UNIVERSIDAD SEÑOR DE SIPÁN

ESCUELA DE POSGRADO

TESIS

METODOLOGÍA INTEGRAL DE CASOS DE PRUEBA SUSTENTADO EN UN MODELO DE VERIFICACIÓN DE REQUISITOS PARA DESARROLLO DE SOFTWARE COMERCIAL

PARA OPTAR EL GRADO ACADÉMICO

DE DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

Autor:

Arangurí García María Ysabel

ORCID

https://orcid.org/0000-0001-9220-5801

Asesor:

Bravo Jaico Jessie Leila

ORCID

https://orcid.org/0000-0001-6841-2536

Línea de Investigación:

Infraestructura, Tecnología y medio ambiente

Pimentel – Perú

2021

UNIVERSIDAD SEÑOR DE SIPÁN

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

"METODOLOGÍA INTEGRAL DE CASOS DE PRUEBA SUSTENTADO EN UN MODELO DE VERIFICACIÓN DE REQUISITOS PARA DESARROLLO DE SOFTWARE COMERCIAL."

AUTOR

Mag. MARIA YSABEL ARANGURI GARCIA

PIMENTEL – PERÚ

2021

“METODOLOGÍA INTEGRAL DE CASOS DE PRUEBA SUSTENTADO EN UN MODELO DE VERIFICACIÓN DE REQUISITOS PARA DESARROLLO DE SOFTWARE COMERCIAL.”

APROBACIÓN DE LA TESIS

_____

Dr. Callejas Torres Juan Carlos

Asesor Metodológico

_____

Dr. Bustamante Quintana Pepe Humberto

Presidente del jurado de tesis

_____ _____

Dr. Callejas Torres Juan Carlos | Dra. Bravo Jaico Jessie Leila

Secretario del jurado de tesis | **Vocal del jurado de tesis**

DEDICATORIAS

Dedicada a Dios todopoderoso por darme la vida
y la oportunidad de la misión encargada. A mi
madre por sus enseñanzas de perseverancia,
honestidad y humildad en la consecución de metas

Dedicada a mi familia por su paciencia, tolerancia,
acompañamiento y lo principal amor que me brindan
siempre a pesar de todo, a Willy, Naomi y Nasya,
artífices de este logro.

A todos mis seres queridos y principalmente
mi asesora qué creyendo en mí, me ha impulsado
a seguir siempre.

AGRADECIMIENTO

Al Dr. Juan Carlos Callejas porque su impulso no sólo a través de su conocimiento, sino también a su apoyo y guía permanente para seguir creyendo que es posible alcanzar este reto.

Un especial agradecimiento al Ing° César Vallejos Dávila, gerente de la empresa que me brindósu apoyo en el proceso de desarrollo de este trabajo de investigación.

A la Dra. Patricia Campos, las autoridades de la Facultad de Ingeniería y colegas de la institución donde laboro, que me animaron a seguir siempre.

RESUMEN

La presente investigación brinda una alternativa de solución a las discrepancias temporales, económicas y de recursos humanos derivadas de la incorrecta interpretación de los requisitos entre lo planteado por el experto en el negocio y lo elicitado por el experto desarrollador del software comercial, que limita la industria del software a las empresas que lo gestionan. Se investigó el proceso de verificación de los requisitos de software. Las causas encontradas se enfocan en mecanismos independientes para evaluar fallas en las fases del ciclo de vida del desarrollo del software, razón por eso se sugiere como objetivo general, aplicar una metodología integral de casos de prueba, basada en un modelo de verificación de requisitos software, con un enfoque sistémico y procesos de retroalimentación continua, para el desarrollo de software comercial.

Como aporte teórico, el modelo de verificación de requisitos en el desarrollo de software comercial con 5 dimensiones, realizado a través del aporte práctico de la metodología integral de casos de prueba, con 6 fases, sugiriendo una adecuada dinámica a lo largo del ciclo de vida, para mejorar la calidad del desarrollo del software comercial, con base en estándares de calidad principalmente la ISO 25000, complementada por la ISO 12207, IEEE 1233-1998, IEEE 1063-2001 y la IEEE 1012-1998, la IEEE 830-1998.

Finalmente, la validación corroborando los resultados de la investigación mediante un pre experimento de la metodología integral de casos de prueba para el desarrollo de software. Para el procesamiento y análisis de datos se utilizó el programa estadístico SPSS.

Palabras clave: Calidad de software, casos de prueba, modelo de verificación de requisitos, metodología integral de casos de prueba proceso de verificación de requisitos software.

ABSTRAC

This research provides an alternative solution to the temporal, economic and human resources discrepancies due to the incorrect interpretation of the requirements between what is stated by the business expert and what is elicited by the expert in commercial software development, which limits the software industry to the companies that manage it. The software requirements verification process was researched. The found reasons focus on independent mechanisms to evaluate failures in the phases of the software development life cycle, this being the reason why it is suggested as a general objective to apply a comprehensive methodology of test cases, based on a software requirements verification model with a systemic approach and continuous feedback processes, for the development of commercial software.

As a theoretical contribution, the requirements verification model in the development of commercial software with 5 dimensions, carried out through the practical contribution of the integral methodology of test cases with 6 phases, suggests adequate dynamics throughout the life cycle improving the quality of commercial software development, being based on quality standards, mainly ISO 25000 supplemented by ISO 12207, IEEE 1233-1998, IEEE 1063-2001 and IEEE 1012-1998, IEEE 830-1998.

Finally, the validation corroborating the research results was done through a pre-experiment of the comprehensive methodology of test cases for software development. The SPSS statistical program was used for data processing and analysis.

Keywords: Software quality, test cases, requirements verification model, methodology of test cases, software requirements verification process.

Índice

I. INTRODUCCIÓN

1.1 Realidad Problemática.

Con el tiempo, la industria del software, ha propuesto combinaciones sinérgicas diversas para la prevención y eliminación de defectos, previos a la fase de prueba, así como pruebas formales aplicadas por personal certificado, pero post implementación, con la finalidad de garantizar un alto nivel de eficiencia en la eliminación de defectos, con costos y plazos reducidos. (Sommerville, 2016)

Es por ello que, el área en la que está ubicada las ciencias de la computación, según (ACM, 2020) Association for Computing Machinery centrada en la línea de investigación de la Ingeniería de Software, plantea parámetros de calidad para medir los aspectos de su proceso de desarrollo.

Según (Sommerville, 2016), conceptualiza la ingeniería de software como una disciplina aplicada con herramientas, técnicas y metodologías en el proceso de producción de software. Sin embargo, destaca que, para evitar disconformidades, es importante hacer uso sistemático y repetitivo de actividades de identificación, documentación y mantenimiento de un conjunto de requerimientos, brindando las pautas necesarias con base a modelos de verificación y validación de requisitos aplicados al seguimiento del desarrollo de software.

El objetivo es satisfacer las necesidades del panorama comercial y estas sean de calidad, lo que indica encontrar un alto nivel de eficiencia en la eliminación de defectos, con costos y tiempos de entrega reducidos para los clientes, en el desarrollo de aplicaciones de software. (Carrizo & Alfaro, 2018)

(Sommerville, 2016) señaló que, a pesar de la magnitud del impacto en los costos, así como en los plazos de desarrollo incumplidos, la fase de verificación y validación, para algunos desarrolladores aún no está prevista de manera formal, dentro de su planificación. Como se señala en el artículo (Xavier, 2015) no es posible, medir la influencia que puede tener una mala especificación de requisitos en el proceso de desarrollo de software, lo que determina el logro de la calidad, en el seguimiento de requisitos específicos, tanto funcionales y no funcionales. Así mismo, para los requisitos en (ISO25000, 2018) indica que es importante que tenga la correcta elicitación e implementación de requisitos, apoyados en guías, modelos, u otras herramientas expresadas en lenguajes formales o no, según el contexto del negocio.

El proceso de verificación de los requisitos software durante el ciclo de vida, permite reducir niveles de inconsistencia, imprecisión con el uso de métodos, que definan características de calidad, para que no se generen excesos en los tiempos de entrega, altos costos de corrección post implementación y el deficiente uso de recursos, a causa de interpretaciones equivocadas de los requisitos para el desarrollo del software. (Pauta Ayabaca & Moscoso Bernal, 2017)

Según el Informe, (Produce, 2016) el 28.3 % de mipymes formales usan algún tipo de aplicaciones software en sus negocios, lo cual ayuda a mejorar aspectos de la gestión administrativa de la empresa, estas aplicaciones responderán a las necesidades de estos usuarios mipymes, en la medida en que se hayan desarrollado, atendiendo a cabalidad a los requisitos establecidos por el experto en el negocio, en contextos en los que se desenvuelven.

En las empresas de desarrollo de aplicaciones software, el proceso de verificación de requisitos se realiza de forma parcial o independientemente durante el ciclo de vida de desarrollo del software (CVDS) (Sommerville, 2016), o especialmente centradas en la etapa de pruebas; pero no de manera integrada, de tal manera que establezca una conexión entrada/salida al término e inicio de la nueva etapa en el ciclo de desarrollo, verificando que los requisitos sean elaborados según lo que el experto en el negocio solicitó y según sus necesidades. Por lo que aún hoy en día se calcula que esta actividad, sino se desarrolla como un proceso de pruebas serio e integrado a lo largo del CVDS, podría requerir un tiempo similar al de la programación para las correcciones pertinentes por los errores encontrados, lo que de acuerdo a las investigaciones previas se evidencia un alto costo económico, que como lo indica Boehm (Boehm, 2006), es aproximadamente un 45% de los errores los que se originan en la etapa "elicitación de los requisitos y en el diseño preliminar", afirma que el 56% de los errores que tienen lugar en alguna aplicación, se deben a una mala especificación de requisitos, y aún ahora como lo indica (Sommerville, 2016) no han mejorado los índices durante proceso de desarrollo de una aplicación software.

Producir aplicaciones software, en nuestro país, tiene un lento desarrollo con respecto de otros servicios de exportación, debido a diferentes variables de influencia, que limitan su desarrollo, entre ellas las de tipo legal. Es decir, la ley no establece con carácter restrictivo, quien debe autorizar o calificar la funcionalidad con calidad del software, como si sucede en el caso de la construcción de una obra civil, en la que son los ingenieros civiles, los legalmente autorizados. Aun así, la digitalización del consumo, según el informe

Perú Service Summit 2020 (PromPerú, 2020), ha impulsado a las pymes a automatizar sus procesos, requiriendo sistemas de información comerciales, con desarrollo de software de calidad.

En términos económicos, para el trabajo de los desarrolladores de aplicaciones de software, se puede decir que, como consecuencia del vacío legal, descrito anteriormente, los usuarios necesitan aplicaciones de software, pero sólo valoran el costo más bajo, no necesariamente la elección de un desarrollador profesional. Por lo que, para el experto en el negocio, que requiere una aplicación software le es indistinto, si lo desarrolla un técnico o un ingeniero especializado en el área. (Deutsch, 1979)

Para los argumentos antes expuestos y para dar soporte a un desarrollo con calidad de la aplicación software, se propone una guía metodológica integral, aplicada en la "verificación de los requisitos de software", que integre con hitos de entrada/salida, basada en casos de prueba, mejorando el seguimiento con un punto de vista integral para la aplicación software.

La empresa desarrolladora de software, Cesly Soft & Bussines Srl, no es ajena al problema del deficiente proceso de verificación de los requisitos software, que generan incumplimientos con respecto a los acuerdos de tiempos de entrega, costos y criterios de funcionalidad, por lo que se ha identificado las siguientes manifestaciones:

- Se identifica más de una interpretación de los requisitos por parte del experto desarrollador del software, frente a los planteados por el experto en el negocio.
- Se emplea tiempo y mayores costos, en la elaboración de una aplicación software, con especificaciones de requisitos errados.

- Los costos de corrección de los errores detectados son más elevados, cuando son identificados, en las últimas etapas del desarrollo de software.
- Altos índices de modificación de requisitos, durante la implementación de la aplicación software.
- Se requiere de recurso humano adicional para cubrir los desfases en tiempo del desarrollo de la aplicación software
- Se hace uso de recursos hardware, sobre utilizados, cuando se elabora la aplicación software.
- Se define un nivel de insatisfacción del experto en el negocio, que requiere la aplicación software, con respecto al producto entregado por el experto desarrollador.
- El proceso para el desarrollo del proyecto, aplica técnicas de pruebas de software con base en su experiencia, en alguna etapa del desarrollo de software.
- La estructura organizacional, de las empresas desarrolladoras de aplicaciones, no es clara en cuanto a los roles y responsabilidades.
- Escasa participación del experto en el negocio, que requiere la aplicación software, en el proceso de verificación de los requisitos funcionales, en el ciclo de vida de desarrollo.
- Se generan problemas de aceptación entre los expertos del negocio que requieren la aplicación software y lo entregado por los expertos desarrolladores.
- El jefe de proyecto desestima la importancia del impacto de truncar, la actividad de verificación de requisitos.

- Se carece de un conjunto de acciones que de manera integral apliquen pruebas de aceptación de software, para que a través de un proceso de verificación se diagnóstique si están implementando los requisitos solicitados.

La situación problemática antes descrita en base a las manifestaciones presentadas resumen **el problema científico**: presentado en esta investigación es que las insuficiencias en el proceso de verificación de los requisitos software, limitan del desarrollo de software comercial, según los instrumentos aplicados, se identifican las siguientes causas:

- La insuficiente integración de los casos de pruebas, en la aplicación del proceso de verificación de requisitos, limita hacer seguimiento del desarrollo.
- Insuficiente referencia teórica sobre cómo se aplican los casos de prueba en el desarrollo de software, de forma integral, como proceso de verificación de requisitos.
- Escaso uso de modelos formales, en las empresas desarrolladoras, que determinen una correcta interpretación por parte del experto desarrollador, con respecto a los requisitos software planteados por el experto en el negocio, debido a que no se aplica un correcto proceso de verificación de los requisitos.
- Los métodos de verificación basados en casos de prueba en su mayoría son estáticos y además, no contemplan un enfoque integral, con procesos de retroalimentación continua, en la verificación de los requisitos durante las etapas del desarrollo software.

- Las organizaciones practican una cultura de desarrollo de aplicaciones software de modo empírico, basada en su experiencia y en sus habilidades prácticas.

En vista de las consideraciones mencionadas anteriormente, se evalúa la oportunidad de profundizar el **proceso de verificación de los requisitos de software**, objeto de la presente investigación.

Con base a lo antes manifestado se pudo encontrar, que en el proceso de verificación de los requisitos se ha desarrollado:

Con respecto a la especificación de requisitos, el proceso de verificación se ha venido aplicando con la intención de comprobar si es que, el sistema cumple con los requerimientos específicos (Sommerville, 2016), pero este proceso se lleva a cabo en cada una de las etapas con diferentes técnicas y no de manera integral, con retroalimentación continua que genere un seguimiento exhaustivo desde la primera etapa hasta la puesta en marcha del producto software.

Según (Deutsch, 1979), se indica que la verificación y validación de requisitos muestra que, el desarrollo de software involucra una amplia gama de actividades de producción donde la posibilidad de error humano es muy alta**.** Porque la identificación de errores puede comenzar a suceder al principio del proceso, cuando los objetivos se pueden especificar de manera errónea o imperfecta, así como en las fases de diseño, diseño y desarrollo. Dado que los seres humanos no pueden trabajar y comunicarse bien**,** el desarrollo de software debe ir acompañado garantía de la calidad.

Según Díaz (Díaz, 2002), con la intención de proponer una alternativa al desarrollo de software de calidad, propuso el uso del concepto de reutilización como metodología para diseñar y desarrollar procesos de software, utilizando la Lógica de causalidad

temporal simple con Modelo de estados no especificados (SCTL-MUS), quien incorpora la formalización del proceso, conjugando diferentes **técnicas de descripción formal** (FDTs), y enfoques iterativos e incrementales al mismo. Sin embargo, debido a las particularidades, se considera que uno de los puntos débiles, la implementación del algoritmo de verificación, muchas veces con base en técnicas de modelado de control, propone reutilizar la información de verificación relevante para los modelos de sistemas aligerando la carga computacional, durante las tareas de verificación. Como resultado, los elementos de software tendrán un nivel de abstracción, como requisitos funcionales y la información de verificación, en comparación con los elementos de abstracción de bajo nivel de, como el código, que, además de parecer simple, es menos rentable y atractiva.

(Zamora, 2011) analiza de los beneficios de adoptar un proceso de prueba en el desarrollo de software, formalmente conocido como "verificación y validación", que según él afecta más que el campo, el desarrollo inmediato del proyecto, sino que también se extiende al resto del negocio. Para diferentes campos de actividad e incluso con clientes finales. Considerar la importancia de lograr "un mayor nivel de eficiencia en el desarrollo de software" ofrece un enfoque holístico que, aunque por sí solo no garantiza la calidad del software que se está implementado, sí lo hace. Contribuye a los objetivos de costo, beneficio, tiempo y calidad para el final producto. Así, se recogen las características clave del proceso de prueba, se recopilan "a través de algunos de los modelos más importantes de madurez y mejora", estructura organizativa, habilidades y perfiles profesionales, para obtener posteriormente la combinación óptima, dentro del proceso V&V, en una determinada organización.

(Gutierrez R, 2011) propone un enfoque de control de calidad, basado en la “automatización de la navegación de las aplicaciones web”, adaptado a los requisitos

funcionales y no funcionales. En el primero, se aplica la “pruebas y análisis para los atributos de calidad” junto con las actividades de **verificación y validación**, tomando en cuenta el desempeño, seguridad, métricas de usabilidad, uso y accesibilidad. En cuanto los requisitos funcionales, se centra en la parte cliente de la arquitectura cliente-servidor, observando los diferentes tipos de **pruebas** y análisis de estado que atraviesa el sistema, a medida que avanza el proceso de navegación de las aplicaciones web en curso. La agregación de los resultados de la evaluación se reporta en un informe generado automáticamente, con diferentes tipos de defectos encontrados, problemas potenciales en los atributos de calidad, preseleccionados allí.

(Hu, Zhuang, & Zhang, 2019) En su investigación analiza que los “métodos de modelado y verificación existentes para el software integrado son insuficientes para los requisitos de seguridad cada vez más importantes”. En este artículo, para cumplir con los altos objetivos de seguridad del software embebido, se propone un marco para modelar y verificar de seguridad del mismo basado en métodos formales y semiformales.

(Zhang & Li, 2020) en su investigación revisó sistemáticamente 950 artículos de **pruebas y verificación** (T&V) de software de control con base en redes neuronales, en dominios críticos para la seguridad, encontraron que “los resultados muestran que la corrección, la integridad, la ausencia de fallas intrínsecas y la tolerancia a las fallas han atraído la mayor parte de la atención de la comunidad investigadora. Sin embargo, se ha invertido poco esfuerzo en lograr la repetibilidad y ningún estudio revisado se centró en la configuración de prueba definida con precisión o en la defensa contra fallas por causas comunes”.

De acuerdo a lo anteriormente analizado, se infiere que aún persisten los **insuficientes referentes teóricos y prácticos**, para el desarrollo metodológico que integren la dinámica de proceso de verificación, para el seguimiento del desarrollo de aplicaciones de software comercial, dado que se aplican técnicas independientes.

El Campo de acción de la investigación es la dinámica del proceso de verificación de requisitos

1.2 Trabajos Previos

A pesar de los avances realizados en la industria del software, con respecto a los estándares orientados a la calidad, la ISO (Organización Internacional de Normalización, 2020) procura garantizar que el software como producto o proceso cumpla con las especificaciones de calidad esenciales que esperan los expertos en el negocio, como el enfoque de proceso del ciclo de vida de la ISO 12207 (INDECOPI, 2006), o la ISO 15504 (Normas ISO, 2018) para la evaluar la madurez. Así como el enfoque del producto de software, con la ISO 9126 (Abud Figueroa, Calidad del Software, 2012) para evaluar características internas y externas o como la ISO 14598 (ISO, 2001) con métricas y requisitos en el proceso de evaluación y la ISO 25000 (Ramírez Ramos, 2018) como evolución de las anteriores; aún persiste la necesidad de establecer los lineamientos para satisfacer la necesidad del cliente, que establece como punto de partida la especificación de los requerimientos al momento de solicitar una aplicación de software comercial que automatice los procesos de su organización.

Esa necesidad ha impulsado la investigación para la propuesta de diversas estrategias que permitan a los desarrolladores de software la mejora continua de los resultados del producto, implementar herramientas de soporte en la

interacción entre el experto desarrollador del software con el experto en el negocio.

Desde sus inicios, el desarrollo de software ha procurado desarrollar con calidad cada una de las actividades, señalada por una metodología en particular. Adicionalmente, se crearon estándares que definieron la característica de calidad, para lo cual se fueron evolutivamente identificando técnicas de validación y verificación de software.

La evolución del desarrollo de software ha sido acompañada a través del tiempo, por los siguientes indicadores: métodos, metodologías, técnicas y estándares que se han ido generando e implementando en la búsqueda de la tan ansiada calidad.

Para Hu et. All (Hu, Zhuang, & Zhang, 2019) en su investigación indica que "los métodos de modelado y verificación existentes para el software integrado son insuficientes para los requisitos de seguridad cada vez más importantes". Por ello se propone un marco de **modelado y verificación de seguridad del software** integrado basado en métodos formales y semiformales. El acreditable es un modelo de seguridad extensible ZMsec (modelo de seguridad Z-MARTE), que extiende Z con elementos de MARTE (Modelado y análisis de sistemas en tiempo real e integrados) y FSA (Autómatas de estado finito), para describir tres dimensiones del software: seguridad casos de uso, estructuras estáticas y comportamientos dinámicos. Se discute un ejemplo de software integrado con ZMV, que ilustra y valida el método de modelado y verificación de seguridad propuesto en este documento.

En la investigación desarrollada por Chao et. all (Chao, Qing, Kiu, Yuwen, & Hailong, 2021) se evalúa la complejidad de los sistemas software, refiriendo que las metodologías científicas aún están en evolución continua. En esta

investigación se aborda sistemas ciberfísicos, que son “reactivos distribuidos, concurrentes, asíncronos y basados en eventos con limitaciones de tiempo ahora en todos los rincones de nuestras vidas, necesitamos métodos sólidos para manejar la complejidad cada vez mayor de sus sistemas de software”. Por ello se planteó un enfoque que garantizaba la seguridad a través la verificación formal. “A partir de los requisitos estructurados, así como el diseño de la arquitectura que se plantea para el sistema, se construyen los modelos de comportamiento, incluidos los modelos Rebeca”. Las propiedades de interés también se derivan de los requisitos estructurados, y luego se usa la **verificación** del modelo para **verificar formalmente sus requisitos** con respeto a las propiedades. Los modelos verificados formalmente se pueden utilizar para desarrollar el código ejecutable. Las asignaciones naturales entre los modelos de requisitos, los modelos formales y el código ejecutable mejoran la eficacia y eficiencia del enfoque.

Para los autores Özakıncı y Tarhan (Özakıncı & Tarhan, 2018) las actividades críticas en el desarrollo de software son, la selección de requisitos, que busca determinar “un subconjunto óptimo de los requisitos (características) del software con el valor más alto para un presupuesto determinado”. Sin embargo, los valores de los requisitos pueden depender unos de otros. Se identifican dependencias de valor que no han sido tomadas en cuenta con los métodos de selección de requisitos existentes, esto trae consigo un nivel de insatisfacción del experto en el negocio y la pérdida de valor como la reputación en los proyectos de software. Es por ello que recomiendan un enfoque de “Selección de requisitos consciente de la dependencia (DARS)” basado en el modelo de programación de enteros lineales (ILP) reduciendo el “riesgo de pérdida de valor al considerar las

dependencias de valor entre los requisitos". Estas dependencias de valor se identifican a partir de las preferencias de los expertos en el negocio para los requisitos. "La validez del DARS se verifica mediante el caso de estudio del mundo real y simulaciones". Demostrando la "reducción significativa en la pérdida de valor cuando se emplea el DARS". Además, el modelo ILP de DARS demostró ser escalable a grandes conjuntos de requisitos (experimentó hasta 3000).

1.3. **Teorías relacionadas al tema.**

1.3.1. Caracterización del proceso de verificación de software y su dinámica

El proceso de verificación de software busca comprobar que la construcción del software está cumpliendo con los requisitos y la funcionalidad, establecidos por el experto en el negocio.

El primer artículo que advirtió la necesidad de realizar pruebas al software fue el de Alan Turing (Turing, 1950). En este documento se discuten varias declaraciones que hasta el día de hoy se conocen como "prueba de corrección". Se define: "algún concepto destinado a evaluar, si un programa está funcionando de manera inteligente." Estos surgen para establecer la necesidad de cumplir correctamente los requisitos definidos para ello, así como certificar si los resultados cumplen con estos requisitos. Turing, determinó que una prueba de software mediría su comportamiento inteligente. Para ello, el funcionamiento del software contrasta con la respuesta de un ser humano, que parece un tercero, que tendrá el rol de evaluador.

Este paso de evaluación, que establece el desempeño de las pruebas, requiere de un esfuerzo de mínimo del 30%" (Zhang, Kitchenham, & Jeffery, 2007) al 50% (GeeksforGeeks, 2021), del costo total del desarrollo del equipo software.

Algunos autores como (Zhang & Wang, 2011) y (Ramamoorthy, Ho, & Chen, 1976) en sus trabajos concluyen que, si el proceso de pruebas es automatizado, reduciría su costo de manera significativa. Es por esto que se determina que el volumen de datos generados para “casos de prueba de software” es de gran importancia para el éxito de la prueba, el porcentaje de software completamente probado es bastante bajo, según la conclusión de Myers et. All (Myers, Badgett, & Sandler, 2012), porque el número de casos de prueba necesarios es infinito. Además de un diseño adecuado, contribuirán a la detección de altas tasas de fallas. Otra forma en que algunos investigadores han "contribuido a la mejora de la supervisión del desarrollo de software” en la década de 1970 es con propuestas metodológicas para automatizar la generación de datos de prueba, con el título "Un sistema formal para probar y depurar programas por ejecución simbólica" en su estudio (Boyer., Elpas., & Levitt., 1975) también presenta la investigación “Un sistema genera datos de prueba y ejecuta programas de manera simbólica.” Y (Clarke, 1976), “Generar datos de prueba automáticamente”, luego (Korel, 1990), y “Generar datos de prueba de programa automáticamente”, (Ramamoorthy C., 1976). De esta forma, se han establecido posibles procedimientos para obtener datos de prueba para aplicar en cada actividad.

(Myers, Badgett, & Sandler, 2012), propuso un modelo orientado a la detección de pruebas denominado “el proceso de ejecución un programa con el objetivo de encontrar errores.” Aquí se establecen los objetivos de detección de fallos de ejecución, asumiendo que se han seleccionado en el software los objetivos que no muestran fallos. Esto no es absoluto, ya que inconscientemente podemos elegir datos de prueba, que pueden no causar problemas de software. Entonces, si queremos demostrar niveles de error, los datos de prueba deben poder detectarlos.

Por lo tanto, la propuesta sigue dos “principios fundamentales en el desarrollo, verificación y validación de software”, incluidas las pruebas de software. El crecimiento de la industria se basó en las pruebas para dar paso a técnicas como la prueba y la evaluación; y estrategias como imágenes en blanco y negro. En el desarrollo de los procesos de verificación, se enfatizó la transición de las pruebas demostrativas a las pruebas de detección, que marcan la propuesta de actividades de detección de errores.

Esto se encuentra en publicaciones como “Testing and Validating Software Projects” (Deutsch, 1979) y la denominada “Software Lifecycle Validation” en (Howden, 1982), se analiza el desarrollo de pruebas y se aplica revisión técnica. En 1982 Bird (Bird & Munoz, 1983) en la publicación “Generación automatizada de casos de prueba aleatorios auto impulsados” propuso la “metodología para automatizar casos de prueba, generado aleatoriamente”, que genera datos aleatorios en cualquier caso de prueba, pero con una tasa de cobertura baja. Donde “cada una de ellos contiene lo desarrollado por su antecesor”. Sigue la filosofía general de la metodología de sistemas de procesamiento de información (FIPS), donde se puede encontrar la siguiente cita “…Ninguna técnica de verificación y validación puede garantizar la precisión”. Sin embargo, la selección cuidadosa de técnicas para un proyecto en particular puede ayudar a asegurar el desarrollo y mantenimiento de la calidad del software del proyecto.

Sin embargo, la definición de "verificar y confirmar" todavía no es clara y precisa. Estos dos términos a menudo se confunden, (Boehm, 2006) afirma la proposición mencionada a través de la pregunta ¿El producto está correctamente construido? "Y confirme mediante" ¿Se está fabricando el producto correcto? Por lo tanto, orientar sus respectivas actividades.

Ese mismo año, un grupo del comité de ingeniería del software IEEE (IEEE, 829-1983 - IEEE Standard for Software Test Documentation, 1991) comenzó a trabajar en un estándar para la documentación de las pruebas del software. Este proyecto no tenía la intención de estandarizar las mejores prácticas implementadas en ese momento, mediante la documentación de experimentos realizados por consenso. Por lo tanto, el documento se ha presentado como un sistema de datos estructurado que debe cumplir con los requisitos de información y acceso del usuario. El resultado fue ANSI/IEEE STD 829-1983 (IEEE, 829-1983 - IEEE Standard for Software Test Documentation, 1991) publicado en 1983 que definió el contenido y el formato de ocho documentos estándar. Estos tuvieron en cuenta aspectos relacionados con la "modularidad, coherencia, acoplamiento, usabilidad y facilidad de revisión de las pruebas."

La principal diferencia entre la propuesta (IEEE, 829-1983 - IEEE Standard for Software Test Documentation, 1991) y las actividades realizadas en ese momento, se mantuvo dentro de las especificaciones de diseño y planificación pruebas. El plan de prueba establecido por esta norma se basa en sus objetivos de identificar riesgos, establecer una estrategia general, definir estructuras tareas, asignar recursos y responsabilidades, cronogramas de desarrollo y obstáculos que este plan pueda crear. Además, proporciona la identificación y descripción de casos especiales y procedimientos de prueba para distinguir entre las especificaciones dadas. Pero los planes de prueba hasta ahora no han incluido tareas de planificación y diseño. Esta planificación se retrasó significativamente, con limitaciones de tiempo en la elección de la estrategia. Además, de distinción entre especificación de caso y especificación de procedimiento, esta norma proporciona una descripción de características del diseño de prueba. Dibujemos una analogía entre probar y definir

la arquitectura del software. Y se centra en organizar conjuntos de pruebas, para establecer su relación directa con los requisitos del software.

Años más tarde, un segundo grupo de IEEE comenzó a desarrollar un estándar para requisitos en (IEEE, 1008-1987 - IEEE Standard for Software Unit Testing, 2009) que los alentó a diseñar pruebas y demostrar sus diferencias clave en comparación con las prácticas más comunes, así que, hasta ahora, “las pruebas unitarias han pasado desapercibidas”. Un año después de que comenzara el desarrollo de estándares hacia las pruebas unitarias, en 1985, sus autores introdujeron el proceso junto con los diferentes niveles de prueba que ya existían, dando como resultado el llamado “proceso de revisión e inspección sistemática". (Hetzel & Hetzel, 1988) define “un sistema de tareas de prueba, productos y roles” como metodología, que crean consistencia y reducen costos, alcanzando los objetivos establecidos en las pruebas. Basado en el “modelo de prevención del ciclo de vida”, que al mismo tiempo del desarrollo de software, establece "una secuencia de actividades como planificar, analizar, diseñar, implementar, ejecutar y mantener pruebas".

En 1989, en (Watts, 1989) ampliaron los principios desarrollados por Deming a la aplicaci{on en la industria del software a través de su trabajo en IBM y SEI. En la década de los 90 se publicó el libro "Software Testing Techniques" de (Beizer, 1990) en el que se presentaba un amplio catálogo de técnicas de prueba. En él, Beizer comentó que "el acto de diseñar pruebas es una de las técnicas más efectivas para la prevención de defectos", ampliando así la definición de las pruebas como concepto de prevención y promoción de errores. Prueba de desempeño en las primeras etapas de desarrollo. Considerando el nivel de formalidad de las pruebas, la definición del proceso V&V en (IEEE, IEEE standard glosasry of software engineering terminology, 1990) es una verificación detallada y se establece como

un proceso para evaluar un sistema o componente, para determinar cuándo el producto de un desarrollo cumple las condiciones establecidas, al inicio del período. Y la validación es el proceso de evaluar un sistema o componente durante o al final del desarrollo para determinar si cumple con los requisitos específicos. En el mismo año nació una técnica. Otra técnica se basa en la automatización de casos de prueba, con la publicación de una metodología objetiva en "Generación automatizada de datos de prueba" (Korel, 1990). Verifique los datos para alcanzar un cierto estado "independientemente del camino tomado". En 1988 (Hetzel & Hetzel, 1988) estableció procesos de prueba como la "planificación, diseño, implementación y ejecución de pruebas" y su entorno. Hasta ahora, las pruebas se han considerado un proceso gestionado, es decir, un ciclo de vida relacionado con las pruebas. (Beizer, 1990).

Pero 1991 fue el año fundamental, se presentó el proyecto iniciado por SEI muchos años después, el Modelo de Madurez de capacidad de Software (SWCMM), fue presentado. Sobre la base del trabajo de Humphrey, que proporciona un punto de referencia para medir la capacidad de las organizaciones de desarrollo de software, en la ejecución de sus diversos procesos, proporcionando una base para evaluar la estrategia de desarrollo.

Los esfuerzos para definir los mecanismos de aseguramiento de la calidad son fundamentales, durante todo el desarrollo de la prueba, su aplicación en el desarrollo de software permite que la técnica represente la revisión final de los parámetros de ingeniería, diseño y codificación. A medida que los clientes comienzan a rechazar productos poco fiables o de bajo rendimiento, la calidad de software mejora, lo que aumenta la carga sobre las pruebas y la ingeniería del software. Existen diferentes perspectivas para determinar la calidad del software.

Desde el punto de vista del cumplimiento de requisitos (Pressman, 2015) define la calidad del software como el cumplimiento de requisitos de rendimiento y funcionalidad bien definidos, basados en estándares documentados y características latentes que esperan los desarrolladores de software (Pressman, 2015)**.** por otro lado, se ocupa de la calidad, que define "la medida en que un sistema, componente o proceso satisface con los requisitos, necesidades y expectativas del usuario". Sin embargo, los objetivos de calidad del producto establecidos determinarán los objetivos la calidad del proceso, ya que la calidad del producto estará estrechamente relacionada con la calidad del proceso. Algunos creen que la calidad se puede lograr definiendo "los estándares y procedimientos de calidad de la organización y los procedimientos para verificar que el equipo de desarrollo cumpla con los mismos". Su argumento es que las normas deben relacionarse con las buenas prácticas, lo que inevitablemente conduce a productos de alta calidad.

Los gerentes de calidad alientan a los equipos a asumir la responsabilidad de la calidad de su trabajo y desarrollan nuevas formas de mejorar la calidad. Por tanto, estas normas y procedimientos forman la base de la gestión de la calidad. Un documento de calidad es un registro de lo que cada subgrupo está haciendo en el proyecto. Esto se estructura en tres actividades principales:

- Para asegurar de calidad, se establece un marco estándares y de procedimientos organizacionales.
- Quality Planning selecciona los procedimientos y estándares adecuados de este marco, para adaptarlo a los proyectos.
- Control de calidad, define y promueve procesos que aseguran que la calidad del proyecto sea monitoreada por el equipo de desarrollo de software.

El proceso de verificación es muy importante, porque a través de él se obtienen los resultados necesarios para poder tomar una decisión.

Al igual que el proceso de fabricación, el proceso de software consta de dos flujos de producción y gestión interrelacionados. (Wang & Wang, 2008). El proceso de producción está asociado al mantenimiento del producto en sí, mientras que el proceso de gestión proporciona los recursos necesarios para el proceso de producción y control.

Finalmente, las tecnologías que gestionan los procesos de producción y operaciones también provienen del medio ambiente. A la hora de desarrollar un proceso software, se deben tener en cuenta una serie de aspectos esenciales señalados por Canfora (Canfora, 2004):

- La tecnología de desarrollo y mantenimiento de software, proporcionan las herramientas y la infraestructura necesarias, para permitir la creación y el mantenimiento de los productos de software complejos, que satisfacen las necesidades actuales y futuras.
- Los métodos y técnicas de desarrollo para el mantenimiento software, constituyen un soporte metodológico fundamental.
- El comportamiento organizacional, es decir, la ciencia de la organización y las personas es útil, en general, los proyectos de software se llevan a cabo en equipos de personas que necesitan ser coordinados y liderados en una estructura organizacional efectiva.
- Economía, porque como cualquier otro producto, el software debe estar orientado a satisfacer las necesidades del experto en el negocio /usuario real.

El proceso de desarrollo adoptado por un proyecto dependerá de sus metas y objetivos. Para lograr estos objetivos, se han desarrollado varios modelos de ciclo de vida, para asegurar la calidad de su desarrollo. Para ello, al final de cada fase del ciclo de vida, es necesario verificar que el trabajo realizado hasta el momento ha alcanzado los objetivos planificados. De esta forma, es más eficaz corregirlos que si se descubren en una etapa posterior. El objetivo final del proceso de verificación y validación es verificar que el sistema está diseñado para un propósito específico al que se aplican las pruebas y la evaluación. Según (Sommerville, 2016) el proceso de validación y verificación (V&V) es "... un conjunto de procedimientos, actividades, técnicas y herramientas, utilizados en conjunto con el desarrollo de software", en cuyo objetivo es asegurar que un producto resuelve el problema original. Luego (Sommerville, 2016) y (Pressman, 2015) especifican que "... la verificación comprueba la consistencia del software con las especificaciones y requisitos", es decir, si responde la pregunta: ¿Se ha construido correctamente el software?

- Proceso para determinar si los productos en una fase del ciclo de vida del software cumplen de forma independiente los requisitos establecidos en una fase anterior.
- Durante este proceso, determine si el producto resultante es completo, consistente y correcto para comenzar con el siguiente paso.
- Validación, comprobando que lo que se ha especificado e implementado realmente coincide con lo que realmente quiere el experto en el negocio, respondiendo a la pregunta: ¿se ha construido el software correcto?
- El proceso de determinar si el software cumple con sus especificaciones.
- El proceso de asegurar que el software generado funciona según lo previsto y coincide con las expectativas del cliente.

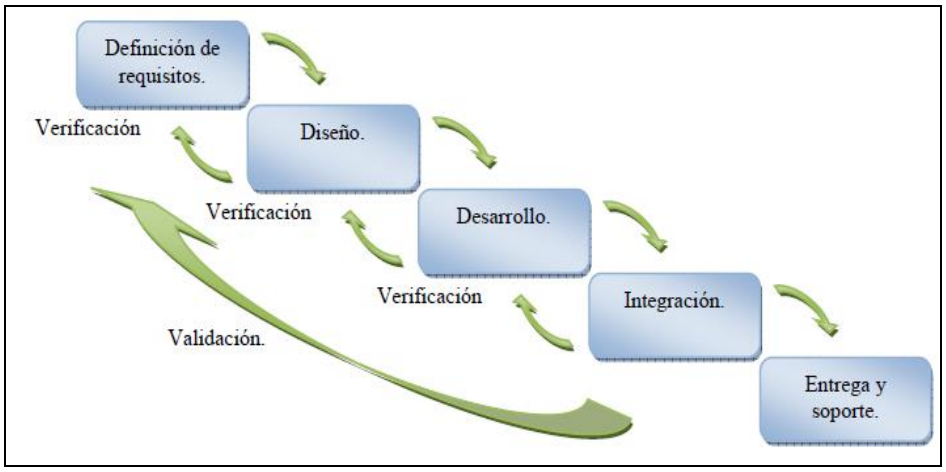


Figura 3. V&V en el ciclo de vida software

Según (IEEE, IEEE standard glosasry of software engineering terminology, 1990) la ingeniería de software proporciona un enfoque sistemático para el desarrollo, operación, mantenimiento y desmantelamiento de software, que, como cualquier disciplina de ingeniería, se define las siguientes características "tecnología" proceso bien entendido, bien definido, resultados predecible de los pasos del proceso de manera repetible.

Por tanto, según (Pohl & Metzger, 2006) ingeniería de software se define "…como producto y sistema. Lo que quiere decir que simultáneamente debe tener un enfoque ingenieril como un enfoque sistémico.". Además, dado que el producto es un sistema debe considerar "…las características de todo sistema: el propósito, globalidad, entropía y homeostasis", de acuerdo a la Teoría General de Sistemas (Von Bertalanffy, 1976). En este enfoque, la especificación de requisitos se enmarca, como parte del proceso de desarrollo de software.

Según (Abud Figueroa, Calidad de la industria del software. La norma ISO-9126, 2012) "…procura comprender y definir correctamente las necesidades que plantea

el experto en el negocio". Para (Bayona-Oré, Chamilco, & Perez, 2019) comprende "…actividades de descubrimiento, modelación, análisis y mantenimiento del conjunto de requisitos identificados", para la elaboración de un producto software. La complejidad de los problemas a resolver exige centrar la atención a su correcto entendimiento antes de emprender la elaboración del software. El término "ingeniería de requisitos" aparecer publicado formalmente en enero de 1990 en (IEEE, IEEE standard glosasry of software engineering terminology, 1990),

Del proceso de Verificación (IEEE C. , 2005) Nótese la importancia de revisar cada producto en producción, ya que se asume que "... si lo que se construye es correcto, también lo es el producto final". Asimismo, se puede observar que "el proceso de Autenticación resalta la importancia de verificar el cumplimiento de los requisitos de la solicitud y del sistema final pretendido. Según (Dorfman & Thayer, 1997) si después de la fase de requisitos aparece un segundo diseño de alto nivel del sistema, también se puede preparar un plan de prueba de integración, el plan se probará después de haber sido (o cuando se hayan) sistematizado los distintos módulos del sistema. Esta correspondencia entre las etapas de desarrollo y los niveles de prueba crea lo que se conoce como el "modelo V", un ejemplo del cual se muestra en la Figura 4.

De hecho, la figura muestra que, en el proceso de desarrollo, hay un punto especial, en las pruebas pasamos de las más específicas a las más generales

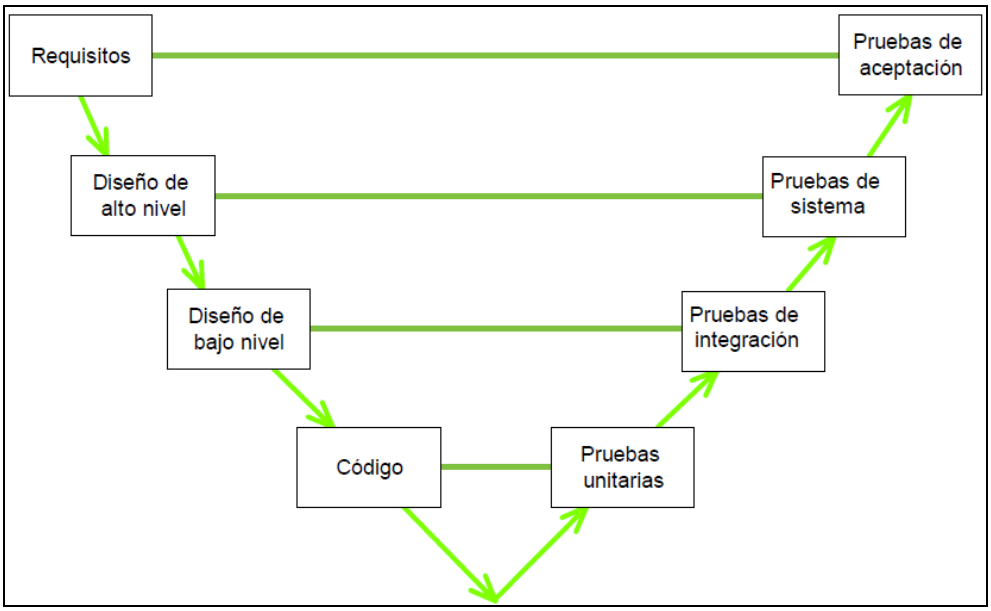


Figura N° 4:

Según la investigación (Díaz, 2002) los metas organizacionales son una buena base para establecer la relación entre las metas que persigue la empresa y los requerimientos del sistema de información a desarrollado", pues todos estos requerimientos (funcionales y no funcionales) deben corresponder a las tareas que se desea realizar en un proceso empresarial. El documento se divide en dos partes principales: (a) la construcción de un modelo comercial a partir del análisis objetivo (b) derivación de un modelo de requisitos de software a partir de un modelo comercial. Este trabajo permite tener un sólido punto de partida sólido para la construcción de sistemas de información, donde cada requerimiento se deriva de los objetivos comerciales.

(Toval Álvarez, 2009), en su investigación sobre la integración de modelos de análisis de dominio y requisitos del lenguaje natural, se construyó de forma incremental en 3 etapas: "(1) Investigando la reutilización del texto de requisitos, que define un método basado en IR con base en la reutilización de requisitos del lenguaje". , denominado SIREN (Simple Reusable Claims), validado en un entorno

industrial. También se ha propuesto "una extensión de SIREN al desarrollo de software global llamado SIRENGsd" (gsd: desarrollo de software global), que se presenta como una colección de amenazas y contramedidas que protegen a los IR cuando se implementan en entornos globalizados, a partir de una revisión sistemática de la literatura (RSL). La segunda fase (2) describe SIRENspl (spl: línea de productos de software) un desarrollo de SIREN con el objetivo de modelar el campo de la línea de productos, sistemas operados a distancia para el mantenimiento de cajas (STO). Integra “técnicas de análisis de dominio específicas y seleccionadas” de la investigación avanzada de IR para líneas de productos e incluye soporte para automatización específica; y finalmente, (3) despertar "interés en integrar modelos de ingeniería de software con especificaciones de requisitos escritas" sobre la base de modelos de ingeniería de software. Este enfoque se ve corroborado por una RSL sobre la correspondencia de adaptar los modelos de análisis de dominio de SIRENspl a los requisitos del lenguaje natural de SIREN. Para evaluar la viabilidad del aplanamiento, existen técnicas de transformación de modelos, "... mediante la descripción formal de los modelos de salida y destino", así como el uso de un lenguaje declarativo de transformación. Finalmente, “lo confirma su aplicación retrospectiva a modelos del caso de estudio de STO”.

En el 2010 se desarrolló una técnica para derivar "... de un conjunto de casos de uso, en forma de una lista de eventos, a una primera especificación formal” escrita en la descripción RAISE de RSL (Lenguaje especial)" en la búsqueda (George & Haxthausen, 2003), que incluye las firmas de funciones de nivel superior del sistema y los tipos representados como tipos. Como entrada, hay casos de uso en forma de listas de eventos. Estos eventos son procesados por un motor de análisis de lenguaje natural para reconstruirlos en un formato estructurado, mediante el cual se

pueden diseñar reglas de transformación para traducirlos en firmas y tipos de funciones en RSL. Estos eventos utilizan un motor de análisis de lenguaje natural para reconstruirlos en un formato estructurado que puede introducir reglas de transformación para traducirlas en firmas de función y tipo en RSL.

Los métodos de prueba de verificación dinámica, se han aplicado de forma independiente a cada fase del ciclo de vida del desarrollo del software, pero no se integran sistemáticamente con procesos de retroalimentación continua, por lo que no existen referencias teóricas y prácticas al respecto.

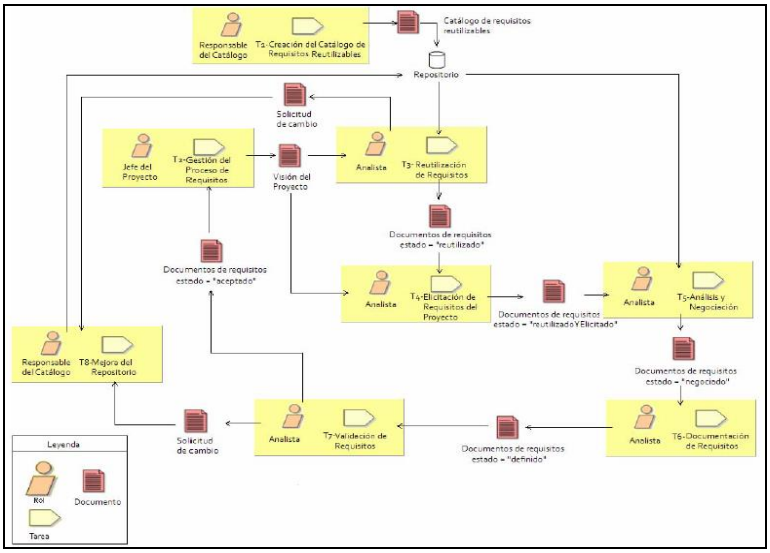


Fig. N° 4: Proceso SIREN

1.3.2. Tendencias históricas del proceso de verificación y su dinámica.

Etapa 1: Crisis del Software (Fines de los 50´ - Década de los 60’s)

La tendencia de usar software de aplicación se nota cada vez más en todas las actividades de la vida humana desde el contexto, personal, social y profesional. Las aplicaciones de software están en todas partes, desde frenos de automóviles hasta controles de hornos microondas. Especialmente, en las medianas y grandes empresas, toman en cuenta la importancia estratégica de sus políticas en materia de

tecnologías de la información y el papel del software en el marco de estas tecnologías, señalado por (Leite, 1997).

Durante 40´s y 50´s, el hardware de uso general era común, en términos de software se adaptaba a cada aplicación y su distribución era limitada. El software no se considera un producto, al igual que los programas se desarrollan para venderse a uno o más clientes. El diseño es un proceso implícito que se realiza en la cabeza de alguien y la documentación a menudo no existe (Pressman, 2015). Las características de las aplicaciones de software para ser considerado exitosas eran: a) se ejecutan, b) se ejecutan rápidamente, c) proporcionar comentarios aceptables y de calidad depende en gran medida de la habilidad del programador.

A fines de la década de 1960, surgieron muchos problemas recurrentes durante el desarrollo de aplicaciones de software, un período conocido como la "crisis del software", debido a la naturaleza repetitiva del proceso de producción, como la entrega, retrasos, presupuestos elevados y necesidades comerciales débiles, así como expertos y dificultades en el uso, mantenimiento y mejora del sistema. (Dorfman & Thayer, 1997) a medida que evoluciona la industria del software, la calidad entra en juego. En 1969, surgió un conjunto de técnicas, conocidas como Ingeniería de Software, en respuesta a esta crisis. Estas técnicas para tratar el software como un producto técnico se enmarcan en fases: "planificación, análisis, diseño, implementación, pruebas y mantenimiento". (Pressman, 2015).

Según (Nauer & Randall, 1969) eleva el significado de la Ingeniería de Software como el establecimiento y uso de principios de ingeniería para obtener en forma económica, software confiable y que trabaje eficientemente en máquinas reales.

Etapa 2: Desarrollo de Software y la ingeniería de requisitos (Década de los 90's).

Durante los siguientes veinte años, hubo discusiones sobre si la creación de software era un arte, una ciencia, o una disciplina (Hoare, 1994). El término fue finalmente, el término se adoptó como "Ingeniería de Software"

En la investigación de (Guezzi, Jazayeri, & Mandrioli, 1991) se define la ingeniería de software se define como "el campo de la informática que se ocupa de la construcción de sistemas de software", que pueden ser tan complejos que deben ser construidos por un grupo o grupos de ingenieros. Según Alan Davis en (Thayer, Dorfman, & Foreword by Davis, 1997) la ingeniería de software es la aplicación de principios científicos para 1) la transformación ordenada de un problema en una solución software, y 2) el mantenimiento de dicho software hasta el final de su vida útil. Para (Pressman, 2015) los objetivos de la ingeniería de software son crear y aplicar 1) una metodología de planificación, desarrollo y mantenimiento orientada al ciclo de vida, bien definida; 2) un conjunto establecido de componentes de software que documentan cada etapa del ciclo de vida y muestran un seguimiento paso a paso y 3) un conjunto de hitos predecibles que se pueden revisar periódicamente a lo largo del ciclo del software.

Etapa 3: Calidad de Software y la Verificación de requisitos (Años 2000)

Dado que se intentó de formalizar el desarrollo de un software, enmarcándolo en el contexto formal de la Ingeniería, esto evidencia la necesidad de considerar a la hora de elaborar estas fases: "análisis de requisitos, estrategias de implementación, modelos de costos, etc".

El Dr. Windston Royce señaló esta situación en (Royce, 1991): “Crear nuevo software que sea amigable para el cliente/profesional y libre de errores es un problema sorprendentemente difícil. Este es quizás el problema más difícil de la ingeniería actual y se ha reconocido como tal durante más de 15 años. La "crisis del software" se considera la más larga del mundo de la ingeniería y aún persiste. A partir de este análisis, es necesario definir un subcampo de la Ingeniería de Software, considerada Ingeniería de Requisitos, en la que se proporcionan métodos, técnicas y herramientas para determinar lo que las personas quieren en función de la aplicación de software generada. En (Thayer, Dorfman, & Foreword by Davis, 1997) la ingeniería de requisitos se define como “la ciencia y la disciplina que se ocupan de establecer y documentar los requisitos de software”. Esto incluye recopilar, el analizar, definir, la verificar y gestionar los requisitos. Según (Kotonya & Somerville, 1998), el proceso de ingeniería de requisitos implica una comprensión clara de los requisitos del sistema deseado. Incluida la participación en un diálogo continuo entre la empresa y el especialista en sistemas, enmarcado en lo que Leite llama, el Universo del discurso (UdeD) incluyendo todas las fuentes de información y todos aquellos involucrados en el software, quienes también son llamados el agente de este universo superior (Leite, 1997). Para establecer un entorno de comunicación adecuado, algunos autores sugieren utilizar enfoques con base en el lenguaje natural; otros tienden a utilizar lenguaje y representación artificiales. Algunos recomiendan crear un vocabulario que capture la jerga utilizada por los expertos en la materia (Anton, Earp, & Alspaugh, 2001) (Arango, Schafer, & Prieto, 1993), la mayoría de los cuales (Benner, Feather, & Johnson, 1993), (Carroll, 1995), (Gough, P., ,Fodemski, F. , Higgins, S., & Ray, S., 1995), y otros. Una de las estrategias sugeridas es también utilizar escenarios, para

asegurar un buen entendimiento y una mayor cooperación entre todos los participantes en el proceso de definición de requisitos, conocidos como interesados. Los ingenieros de requisitos deberán comprender, modelar y analizar el dominio de la aplicación en el que se utilizará el software, y los profesionales comerciales confirmarán si las opiniones de los ingenieros son correctas (Hadad, Doorn, Kaplan, & Leite, 1999).

Etapa 4: Procesos de verificación de requisitos Software y su dinámica (En la actualidad)

En (Leite, 1997) se define que el problema de la calidad, se refiere a la confiabilidad de las especificaciones presentadas en forma de escenarios. En el primer tema, se presenta una innovadora estrategia de mediación innovadora que sistematiza el proceso de construcción utilizando relaciones tipificadas y experiencia operativa. Para la calidad, se proporcionan políticas y procedimientos para detectar defectos y errores en situaciones.

En las últimas investigaciones orientadas a la mejora del seguimiento de software, se están evaluando técnicas que apliquen conceptos con enfoque sistémico, es decir bajo la evidencia de las fases del proceso de desarrollo, están en estrecha relación unas con otras, es necesario evaluarlas desde un enfoque integrador y contextualizado al objetivo funcional para el cual están siendo desarrollado.

Además, la búsqueda constante de que estas herramientas tengan una connotación de desempeño inteligente y automatizado, para garantizar la precisión de los resultados.

1.3.3. Marco Conceptual.

Análisis de requerimientos: El nombre correcto de la fase es "análisis de requisitos" o "análisis de requerimientos". El objetivo principal de esta fase es percibir, comprender e interpretar los distintos requisitos para la construcción de sistemas de software.

Calidad de uso: De acuerdo con la norma ISO-9126, esta es la calidad percibida por los usuarios de la aplicación durante la fase de operación y mantenimiento de dichas aplicaciones, la cual está determinada por los atributos que presenta el sistema para definir cualidades externas, expresadas en su desempeño y cualidades internas que son aquellas que el sistema exhibe estáticamente.

Capability Maturity Model (CMM): Modelo de madurez presta especial interés a las pruebas. Las áreas de este modelo que son más relevantes para la fase de pruebas son la ingeniería de producto software, los programas de capacitación, la gestión de cambios y la gestión de cambio de procesos. CMM establece cuatro niveles de pruebas: de unidad, de integración, de sistema y de aceptación, que, además de la de regresión, se realiza para verificar la corrección de los cambios en el sistema. El plan de pruebas debe escribirse y definir los criterios de prueba junto con el resto del proceso. El equipo de prueba debe distinguirse claramente del equipo de desarrollo y las pruebas deben realizarse independientemente de ese equipo.

Elicitar: La elaboración de requisitos incluye por un lado el contexto del sistema y, por otro, el origen de los requerimientos.

Ingeniería de Software: Según Pressman (2010) es una rama o campo de la informática que proporciona “métodos y técnicas para desarrollar y mantener software de calidad”, resolviendo todo tipo de problema. La IEEE Computer

Society IEEE, (1990), la define como la adopción de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento de software.

Ingeniería de requisitos: Es la aplicación de principios, métodos, técnicas y herramientas con el fin de descubrir los requisitos de un producto software, así como el análisis y documentación de sus objetivos, funciones y limitaciones. Mecanismo de los sistemas antes mencionados; sin embargo, existe una falla, es decir, no existe consenso sobre el lenguaje, método o herramienta para hacerlo como se indica (A. &. S. E. Alarcón, 2008). Según Ackoff "fallamos más a menudo porque resolvemos el problema incorrecto, que porque obtenemos la solución incorrecta para el problema correcto."

Método: La organización lógica de diversas técnicas y procedimientos para llevar a cabo su desarrollo. Los métodos definen el orden en el que se deben aplicar las técnicas, los requisitos que se deben proporcionar (documentos, informes, informes, etc.), los controles para ayudar a garantizar la calidad y las pautas para ayudar a desarrollarlos. El gerente evalúa el progreso de las actividades.

Prueba: Las pruebas de software "implican en la verificación dinámica del comportamiento de un programa", sobre un conjunto finito de casos de prueba.

Pruebas Unitarias: Se aplican durante la construcción del sistema, para evaluar el diseño y funcionalidad de los componentes construidos.

Pruebas de integración: Se aplican durante la construcción del sistema, verificando la correcta alineación de los componentes entre ellos a través de interfaces, y si se ajustan a la funcionalidad establecida.

Pruebas de Sistema: se aplica durante la construcción del sistema

(componentes completos). Ellos prueban minuciosamente el sistema, verificando su funcionalidad e integridad general, en un entorno lo más cercano posible al entorno de producción final.

Requisito: son esencialmente todos los elementos y características requeridos, necesarios o deseados por el experto en el negocio.

Software: este es un producto diseñado y construido por ingenieros de software. Esto incluye programas que se ejecutan en computadoras de cualquier tamaño y arquitectura, documentos que incluyen formularios impresos y virtuales, y datos que combinan números como texto, así como representaciones numéricas, audio, video e información de imágenes.

Técnicas de pruebas de integración, Arriba-Abajo: Es el primer componente que se probará, es el primero en la jerarquía. Una de las ventajas es que la interfaz entre los diferentes componentes se prueba pronto y con frecuencia.

Técnicas de pruebas de integración, Abajo-Arriba: los componentes de nivel inferior se prueban primero. Este enfoque permite un desarrollo paralelo, pero provoca una mayor dificultad en la planificación y la gestión.

Validación: El proceso de evaluar un sistema o componente, durante o al final del desarrollo, para determinar si cumple con requisitos específicos.

Verificación: Proceso de evaluación de un sistema o componente para determinar si los productos de una etapa dada cumplen con las condiciones establecidas al comienzo de ese período, independientemente del código actual.

1.4 Formulación del Problema.

El deficiente proceso de verificación de los requisitos software, limita el desarrollo de software comercial.

1.5 Justificación e importancia del estudio.

Según Estrada (2002), la verificar los requisitos de software, se han desarrollado encuestas con el objetivo de apoyar el proceso de desarrollo de software, reduciendo las condiciones de insatisfacción de cada autor participante. La investigación actual en el campo de la ingeniería de requisitos busca mecanismos que permitan establecer una relación entre las funciones esperadas de un sistema de información y los procesos de negocio que soportará. Este enfoque asegurará que el sistema de información desarrollado sea verdaderamente útil en las tareas de los actores organizacionales. La investigación en esta área ha determinado que los objetivos organizacionales son una buena base para establecer una relación entre los objetivos que persigue la empresa y los requisitos de los sistemas de información desarrollados, ya que todos estos dos requisitos (funcionales y no funcionales) deben corresponder a la tarea que desea realizar como parte del proceso empresarial. A su vez, los procesos comerciales permiten el cumplimiento o satisfacción de uno o más objetivos comerciales. En este trabajo, se presenta una propuesta para derivar los requisitos de software a partir de los modelos de negocio. El documento se divide en dos partes principales: (a) construcción de un modelo comercial a partir de un análisis objetivo (b) derivación de un modelo de requisitos de software a partir de un modelo comercial. Este trabajo nos permite tener un punto de partida sólido para la construcción de sistemas de información, donde cada requerimiento es impulsado por objetivos comerciales.

Según (Toval Álvarez, 2009) en las primeras etapas del desarrollo del sistema, se combinaron dos enfoques para especificar un sistema: primero utilizando

una técnica no rigurosa, como casos de uso y listas de eventos, y luego cambiando de ellos a una especificación formal mediante un método formal RAISE. Como entrada, hay casos de uso en forma de listas de eventos. Estos eventos son procesados por un motor de análisis de lenguaje natural para reconstruirlos en un formato estructurado que puede introducir reglas de transformación para traducirlos en firmas y tipos de funciones en RSL.

Por lo antes mencionado se propone un modelo de elicitación de requisitos a partir de la manifestación de sus requerimientos por el experto en el negocio (EN) en lenguaje natural, este modelo sería de utilidad a los desarrolladores de software como una herramienta que garantice la construcción correcta de las aplicaciones requeridas, que podría ser ubicada en la nube, para que a modo de "alquiler", se ofrezca a las Mypes, con bajos costos de inversión por el producto, el mantenimiento y la infraestructura requerida, para lograr un eficiente uso de las TICs.

Se tiene entonces como **aporte teórico** el modelo de verificación de requisitos con base en estándares de calidad de software.

Y como **aporte práctico** se considera la metodología integral de casos de prueba a través de plantillas de retroalimentación continua.

La **novedad científica de la investigación,** radica en revelar la lógica de la integración de métodos de caso de prueba a través de plantillas de retroalimentación continua aplicadas en las etapas del desarrollo del software comercial en la mejora de su seguimiento.

La significación práctica, radica en el impacto social al contribuir en la mejora del seguimiento del desarrollo de software comercial, ya que la guía metodológica de casos de prueba a través de plantillas de retroalimentación

continua basado en un modelo de verificación de requisitos, reduciría los niveles de excedentes en tiempos de entrega establecidos con el cliente, costos de corrección post implementación y el eficiente uso de recursos en el desarrollo de software, frente a las interpretaciones equivocadas por parte del experto en sistemas con respecto a los requisitos planteados por el experto en el negocio.

1.6 Hipótesis

1.6.1 Hipótesis.

Si se aplica una metodología integral de casos de prueba, con base en un modelo de verificación de requisitos software, que tenga en cuenta la relación de la integración de métodos de caso de prueba, a través de plantillas de retroalimentación continua en las etapas de elaboración, entonces se contribuye al desarrollo de software comercial.

1.6.2. Variables.

Variable independiente:

Metodología integral de casos de prueba a través de plantillas de retroalimentación continua, basado en un modelo de verificación de requisitos software.

Variable dependiente:

Desarrollo de software comercial.

1.7. Objetivos

1.7.1. Objetivos General

Aplicar una metodología integral de casos de prueba, sustentado en un modelo de verificación de requisitos software, con un enfoque sistémico aplicando procesos de retroalimentación continua, para desarrollo de software comercial.

1.7.2. Objetivos Específicos

- Caracterizar epistemológicamente el proceso de verificación de requisitos software y su dinámica.

- Caracterizar las tendencias históricas del proceso de verificación de requisitos software y su dinámica.

- Caracterizar el estado actual de la dinámica del proceso de verificación de requisitos software, en una empresa desarrolladora de Chiclayo.

- Elaborar el modelo de la sistematización epistemológica del desarrollo del software comercial.

- Elaborar una metodología integral de casos de prueba a través de plantillas de retroalimentación continua, basado en un modelo de verificación de requisitos software, aplicado al desarrollo de software comercial.

- Validar corroborando los resultados de la investigación mediante un pre experimento.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación.

La presente investigación es de tipo mixta, como lo indica en libro (Hernández Sampieri, Fernández Collado, & Baptista Lucio, 2014) es “Un conjunto de procesos de investigación sistemáticos, experimentales y críticos” implica simultáneamente la recolección, análisis de datos cuantitativos y cualitativos, así como su integración, como discusión general. Inferir como resultado de toda la información recopilada (inferencia sintética) y comprender mejor el fenómeno en estudio.

El diseño de contrastación de hipótesis es cuasi experimental, dado que se realizarán dos experimentos con los datos, el primer experimento determinará el nivel de aceptación de los requisitos software por parte del experto del negocio, con las estrategias actuales y el segundo experimento se realizará con la metodología integral de casos de prueba a través de plantillas de retroalimentación continua, basado en un modelo de verificación de requisitos software.

2.2. Población y muestra.

Para la presente investigación se ha considerado como población a los expertos desarrolladores que le permita hacer un seguimiento del software comercial, así como el experto en el negocio, que permitirán evaluar el nivel de aceptación de los requisitos software, los cuales están distribuidos en 2 categorías principales, experto en el negocio y el experto del sistema.

Categoría	N°
Experto en el negocio	10

Experto desarrollador de software	32
Total	**42**

Tabla N° 01: Muestra poblacional

Al ser una población pequeña, se tomará en cuenta todo el personal como la muestra de la investigación.

2.3. **Técnicas e instrumentos de recolección de datos, validez y confiabilidad**.

Los métodos de investigación utilizados en la investigación son: deducción, razonamiento inductivo, análisis histórico, estructura del sistema funcional y dialéctica holística, para determinar las características del contexto teórico e historia de los requisitos del proceso de verificación en software comercial. Además de la regla general, describir el estado actual del proceso de verificación de reclamaciones en software comercial. Las herramientas de recolección de datos utilizadas en el estudio son: cuestionario, entrevista, observación y análisis de documentos.

Cuestionario: Herramienta de investigación que consta de una serie de preguntas y otras instrucciones diseñadas para recopilar información de los encuestados.

Entrevista: Intercambio de ideas u opiniones a través de una conversación que tiene lugar entre una, dos o más personas a las que el entrevistador es la persona designada para hacer preguntas. Los entrevistadores utilizan técnicas de reunión a través de un interrogatorio estructurado o una conversación completamente abierta; se utiliza un formulario o esquema con preguntas o preguntas para centrar la presentación, que sirve como guía.

Observación: Se observará el desempeño de la red actual y su monitorearla con herramientas permite un análisis más preciso.

Análisis documental: La información analizará y procesará la información que se encuentre en la investigación en Internet, será analizada y procesada, como bases de datos científicas, artículos científicos, información de fuentes confiables y libros recopilados que han contribuido a la presente investigación.

2.4. Procedimientos de análisis de datos.

La información recolectada a través de los instrumentos, será validada, luego codificada, y su tratamiento para su posterior análisis se realizará haciendo uso de los programas SPSS y Microsoft Excel.

Los resultados de este análisis serán presentados a través de gráficos y tablas estadísticas.

2.5. Criterios éticos

Los principales principios éticos que se tomará en cuenta en la presente investigación son:

Secreto profesional: El investigador tiene el derecho y el deber de mantener el secreto profesional sobre todos los hechos y noticias de los que tenga conocimiento como consecuencia de su actividad profesional, con algunas excepciones, justificables ética o legalmente.

Función social: El investigador siempre debe tener presente la naturaleza de su función de servicio social. No debe hacer planes o decisiones que puedan ser antisociales.

Protección a las personas: La persona en todos los estudios es un fin, no un medio, por lo que necesita un cierto grado de protección, que se determinará en función del riesgo que le concierne, en el que incurre y la probabilidad de obtener un beneficio.

2.6. Criterios de Rigor científico.

Integridad científica: La integridad o integridad no solo debe regir la actividad científica de un investigador, sino también extenderse a su práctica docente y profesional. La integridad de un investigador es particularmente relevante cuando, con base en sus propios estándares éticos, se evalúan e informan los posibles daños, riesgos y beneficios que pueden afectar a los participantes de la encuesta. Asimismo, se debe mantener la integridad científica al declarar conflictos de interés que pudieran afectar la realización de un estudio o la comunicación de sus resultados.

III. RESULTADOS

3.1. Resultados en Tablas y Figuras

El diagnóstico de la realidad actual con respecto a las condiciones actuales de desarrollo de software comercial, tomó en consideración tres dimensiones, la primera denominada calidad en la verificación de los requisitos, la segunda ciclo de vida en el desarrollo de software y la tercera gestión del seguimiento del software, se formularon preguntas abiertas y cerradas en un cuestionario. Se aplicó a 32 expertos desarrolladores de software.

Como se muestra en la Tabla 2, con respecto a la dimensión: **calidad en la verificación de los requisitos**, en la pregunta en la que se le solicitaba la descripción del procedimiento que realizan, para la toma de requisitos respondieron un esquema de trabajo propio, es decir con base en la práctica o la experiencia, las cuales se han agrupado en tres etapas de un tradicional ciclo de vida:

ETAPAS	ACTIVIDADES
PRIMERA	Entrevistas definiendo los temas a tratar y con las preguntas pertinentes a cada uno de los stakeholders identificados, con la finalidad de: - Conocer la organización a través de documentación histórica, de procesos y gubernamental proporcionada.

	- Identificar las reglas del negocio, los requerimientos, alcances, expectativas, fechas de entregables y limitaciones del proyecto. - Roles involucrados en el proyecto.
SEGUNDA	Actividades de análisis de la información, en las que se desarrollan: - La identificación de requerimientos funcionales y no funcionales, analizando los procesos, objetivos que permitan minimizar riesgos, estableciendo el alcance del producto. - Revisión de sistemas similares al solicitado. - Evaluar el impacto con otros objetos del sistema del cual depende el desarrollo. - Analizar los requerimientos establecidos, a través del feedback en reuniones establecidas con el cliente. - Evaluar tecnología y herramientas más afines a la solución.
TERCERA	Corresponde a la implementación del producto software, con actividades repetitivas que involucran - Priorización de los requerimientos en base a criterios establecidos. - Validación de requerimientos al final del proyecto. - Determinar la existencia del QA. - Reajustar fechas, requerimientos modificados y estimaciones. - Presentar entregables con detalles del software, tecnología y herramientas. Incluye un diagrama conceptual y uno de despliegue a fin de obtener una revisión preliminar por parte de los interesados.

Tabla N° 02: Actividades por etapa de desarrollo de software

En esta misma dimensión se solicitó identificar los requisitos que con más frecuencia son solicitados por el experto en el negocio, para el desarrollo de un software comercial, cuyas respuestas se clasificaron como se indica en el gráfico 1, agrapándose técnicamente en Funcionales y No funcionales, así como la

subclasificación de estos según (Sommerville, 2016) en el caso de los requerimientos funcionales:

-Del Usuario y

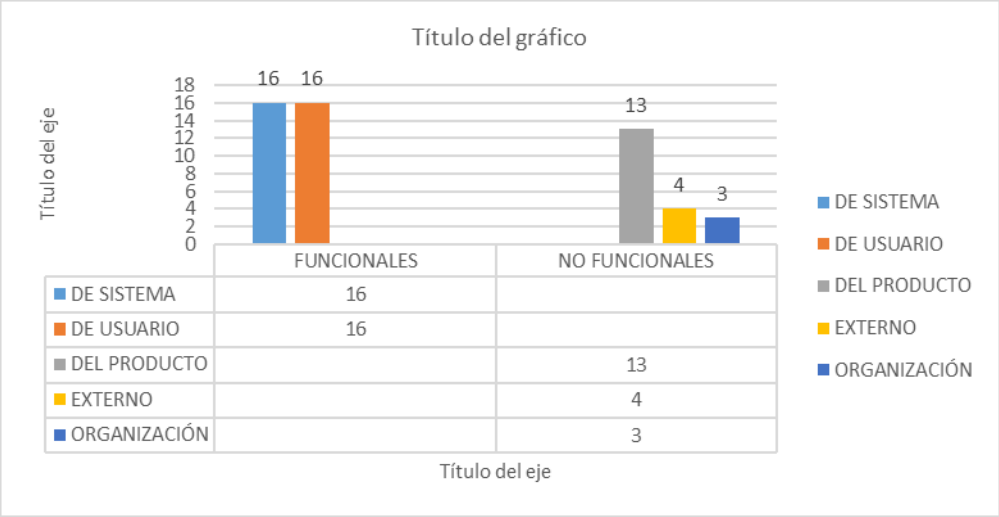
- Del Sistema

Identificándose un 50% del usuario y 50% del sistema. Se encontraron también en las respuestas requerimientos no funcionales que según (Sommerville, 2016), se clasifican en:

- Producto.
- Organización
- Externo

Concluyendo que los requerimientos no funcionales, se centran principalmente en el producto.

Gráfico N° 01: Porcentaje proporcional de requerimientos funcionales y no funcionales



Así mismo en la pregunta, en la cual se busca conocer cuáles son los términos utilizados por los usuarios para describir sus requerimientos, planteados en su lenguaje natural cuando solicita la elaboración de una aplicación software

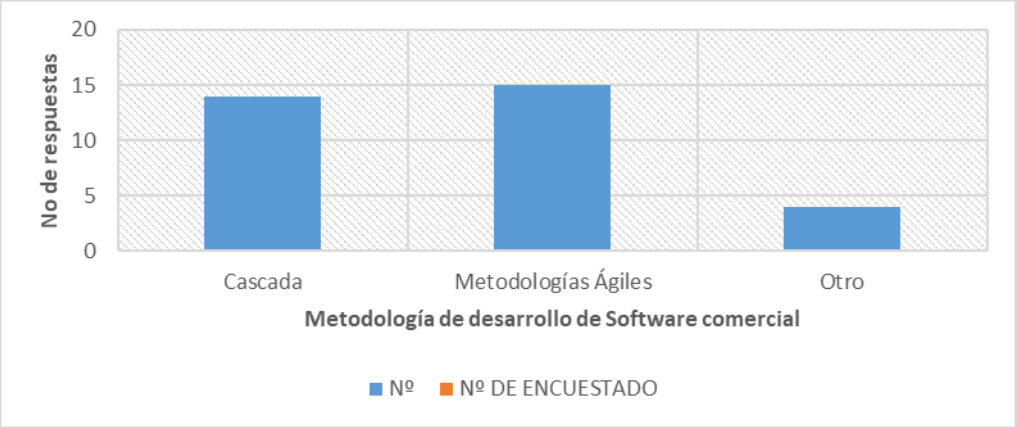
comercial, se señalan según se indica en la Tabla 3. Siendo los procesos o módulos identificados típicamente ventas con 33 respuestas, facturación y almacén con 22 respuestas cada una, compras 8, otros diversos no coincidentes 18, como se aprecia a continuación:

PROCESO O MÓDULO	No DE RESPUESTAS
VENTAS	33
ALMACEN	22
FACTURACIÓN	22
COMPRAS	8
OTROS DIVERSOS	18

Tabla N° 03: Número de procesos solicitados

La segunda dimensión denominada ciclo vida de desarrollo software, se les planteó la pregunta, si formalmente aplican una metodología específica, a lo que respondieron 15 desarrolladores que hacen uso de metodologías ágiles, 14 aún mantienen la tradicional metodología en cascada y otras 4 diversas otras no tan comunes, de los 33 desarrolladores encuestados, de Lima y provincias como se aprecia en el gráfico 2.

Gráfico N° 02: Número de metodologías de desarrollo frecuentemente utilizadas.



En la dimensión denominada **gestión del seguimiento del software** con la pregunta que, busca conocer con qué frecuencia los requerimientos del usuario no son interpretados correctamente por el experto desarrollador de la misma manera, como las plantea el experto en el negocio y que incidencias genera, las respuestas fueron como se aprecia en la Tabla 3, agrupando las frecuencias en alta, media y la que no indica una frecuencia precisa, suman el 72% y solo un 28% presentan frecuencia baja de con las que se presentan estas inconsistencias. La frecuencia de incidencia presentada repercute en modificaciones permanentes, que traen consigo ampliaciones de tiempos de entrega desfasadas a las inicialmente establecidas con el experto en el negocio, costos y recursos adicionales, afectando el nivel de confianza entre el experto en el negocio y el experto desarrollador de software comercial.

NÚMERO DE RESPUESTAS SEMAFORIZADA	NIVEL DE FRECUENCIA	PORCENTAJE
9	Frecuencia alta	28%
7	Frecuencia media	22%
9	Frecuencia baja	28%
7	No indica frecuencia	22%
TOTAL		100%

Tabla N° 04: Nivel de frecuencia de requerimientos interpretados equívocamente por el experto desarrollador de software.

Por lo anteriormente analizado, en otra pregunta se consultó con respecto a las causas que generan este nivel de frecuencia de la incorrecta interpretación de los requisitos solicitados por el experto del negocio, que reduce la claridad de los requerimientos identificados por el desarrollador del software comercial, como lo muestra la Tabla 5, en la que observamos una escasa participación del experto en el negocio con un 22%, interpretaciones equivocadas del analista

desarrollador de software comercial en un 38%, la falta de una actividad formal de verificación de requisitos 22% y la falta de planificación de casos de prueba con un 18%.

Estas manifestaciones que se muestran en la Tabla 5., son algunas de las motivaciones que impulsan la presente investigación, cuya finalidad es aportar con una metodología integral de casos de prueba para el desarrollo del software comercial.

CAUSAS DE LAS INCORRECTAS INTERPRETACIONES DE LOS REQUISITOS	PORCENTAJE
El usuario no tiene tiempo para apoyar en la revisión de requisitos.	22%
El analista no interpreta correctamente los requerimientos planteados por el usuario.	38%
No se establece una actividad formal de verificación y validación de requisitos.	22%
No se planifican los casos de prueba en la implementación de la aplicación.	18%
TOTAL	100%

Tabla N° 05: Causales de una incorrecta interpretación de requisitos por parte del experto desarrollador de software.

Luego de analizar de los resultados obtenidos de las encuestas, evaluamos que:

-Al momento de definir la especificación de requisitos funcionales por proyecto se emplean hasta 10 días adicionales para validar especificaciones funcionales mal interpretados por el experto en el sistema.

-El porcentaje de margen de error en tiempos de entrega por proyecto es hasta el 40%, que es la demora entre el tiempo planificado en el acuerdo de entrega del proyecto y el tiempo efectivo de culminación.

- El porcentaje de tiempos de entrega por correcciones en el proyecto se ha medido en un 10% de fluctuación por el tiempo adicional que se extiende el proyecto por correcciones de requerimientos funcionales mal interpretados.
- Porcentaje de costos adicionales en recursos humanos de ejecución por recursos humanos planificados, hasta un 40% adicionales en la contratación de recursos humanos adicionales para disminuir el retraso de los tiempos de entrega.
- Porcentaje de recursos tecnológicos utilizados por recursos tecnológicos planificados hasta un 20%, de costos adicionales por compra de recursos tecnológicos adicionales para disminuir el retraso de los tiempos de entrega.

3.2. Discusión de resultados

De lo analizado previamente, se infiere que aún al momento en el que se desarrolla la presente investigación, el nivel de incidencias por interpretaciones incorrectas para un mismo requisito por parte del experto desarrollador, expresados por el experto del negocio en lenguaje natural, quien hace uso de una diversidad de palabras sinónimas, que al ser utilizadas generan escasa claridad en el establecimiento de los mismos para el desarrollo del software comercial con un 38% de incidencia, así como la carencia de un plan formal de actividades de verificación de requisitos en un 22% y la falta de planificación de casos de prueba con 18% de incidencia, que cuando se realizan son aplicadas de manera independiente, en las etapas del desarrollo del software comercial.

Que los mayores requisitos están centrados en el módulo de VENTAS con 36 requisitos identificados, por lo que se podría señalar como el principal para el software comercial, es decir el de mayor interés y del cual el experto del negocio conoce más a detalle. Luego el proceso de ALMACEN con 22 requisitos solicitados analizando el interés de tener controlados los productos que administran con respecto a su registro, despacho y reposición, en el mismo nivel de interés calificado por el número de requisitos en número de 22, está el proceso de FACTURACIÓN, para llevar el control de documentos de pago, caja y bancos, en la contabilidad de la empresa.

Finalmente, podemos concluir que el número de manifestaciones etiquetadas como frecuencia baja sólo se señala con un porcentaje acumulado del 28%, señalando escenarios en los cuáles se han presentado un mínimo índice de interpretaciones erróneas, quedando un 72% de incidencias que generan permanentes modificaciones, por las diferentes causales analizadas que se centran en una equivocada interpretación de requisitos que determinan una escasa claridad al momento de la implementación del software comercial, generando altos costos de sobretiempo y recursos utilizados adicionales. Siendo estos los resultados que se utilizan para contrastar la propuesta de tesis

3.3. Construcción del Aporte teórico

Introducción

En esta sección se explica la estructura epistemológica del modelo de verificación de requisitos en el desarrollo de software comercial, adoptando un enfoque integral de cinco dimensiones: dimensión del experto en el negocio, la dimensión de la ingeniería de requisitos, la dimensión del experto desarrollador de software, la dimensión de técnicas de pruebas y la dimensión del gestor de

calidad de software, todas ellas retroalimentándose continuamente, con una significativa participación del experto en el negocio.

3.3.1. Fundamentación del aporte teórico del Modelo de verificación de requisitos en el desarrollo de software.

En el modelo de verificación de requisitos para el desarrollo de software comercial, se considera como uno de los actores al **experto del negocio**, definido por (Pradel Miquel & Raya Martos, 1981) como "… las personas que conocen del dominio del sistema que se está desarrollando" y aquel que da conocer sus necesidades en una **lista de requisitos en lenguaje natural,** insumo solicitado para la elaboración de la aplicación de software, estos **requisitos,** que según (Somerville, 2011) indica que "es una declaración abstracta de alto nivel sobre un servicio prestado por el sistema o como una limitación del mismo" **,** en este caso con base en procesos comerciales, de tal manera que agilice tiempos, administre con eficiencia los recursos. Esta actividad de elicitar los requisitos se desarrolla en la **ingeniería de requerimientos** definida por (Sommerville, 2016) como "…el proceso de descubrir, analizar, documentar y verificar los servicios", obteniendo así los **requisitos funcionales** subclasificado como del sistema y los del usuario. En este proceso quien organiza los requisitos solicitados el **experto desarrollador de software**, quien en su rol de analista-programador, se encarga de recepcionar los requisitos elicitados, registrarlos y **listar los requerimientos priorizados**, adicionalmente se genere el **diccionario de frases** de sinonimias que el experto en el negocio utiliza para referirse al mismo proceso, es decir consolidar las diferentes maneras de expresar sus requisitos de tal manera que se reduce la incorrecta interpretación o discrepancias generadas. Estas discrepancias, se identifican al implementar **técnicas de pruebas**, que según

(IEEE, IEEE standard glosasry of software engineering terminology, 1990) indica que "… usualmente son consideradas para validación; pero también para verificación", y sobre las pruebas de software en su estudio (Myers, Badgett, & Sandler, 2012) las define como "el proceso de ejecutar un programa para encontrar errores". En la **dimensión técnica de prueba** se establece el mecanismo de los **casos de prueba**, que es diseñado para ejercer una ejecución particular o para verificar la conformidad con un requisito específico (Sánchez Peño, 2015). Los resultados de estos casos de pruebas son evaluados en la **dimensión gestor de calidad de software**, siendo aquel que aspira a desarrollar una «cultura de la calidad» donde todos seamos responsables del desarrollo de un producto con un alto nivel de calidad, reconociendo aspectos intangibles en la calidad del software. Ellos ayudan a la gente interesada a medir el indicador de **nivel de impacto de la discrepancia**, la cual, al ser priorizada, dará cuenta que una lista ordenada de desarrollo, implementando una gestión **de alertas**, permitiendo que el experto en el negocio verifique para corregirla, evitando que pasen a posteriores etapas con mayores costos de corrección.

3.3.2. **Descripción argumentativa del Modelo de Verificación de requisitos**

En esta investigación se desarrolló la dinámica del modelo de verificación de requisitos software aplicado a pequeñas y medianas empresas, tomando en cuenta principalmente los casos de prueba aplicada a la gestión de calidad de software, con plantillas de retroalimentación continua reduciendo el índice de discrepancia, a través de la retroalimentación con experto en el negocio.

Para la elaboración del modelo propuesto se ha tomado en cuenta 5 dimensiones: la dimensión del experto en el negocio, la dimensión de la ingeniería de

requisitos, la dimensión del experto desarrollador de software, la dimensión de técnica de pruebas y dimensión gestor de calidad de software.

Además, se plantea de manera transversal la permanente retroalimentación, lo que permite una visión integral con visión sistémica, como se observa en la siguiente figura:

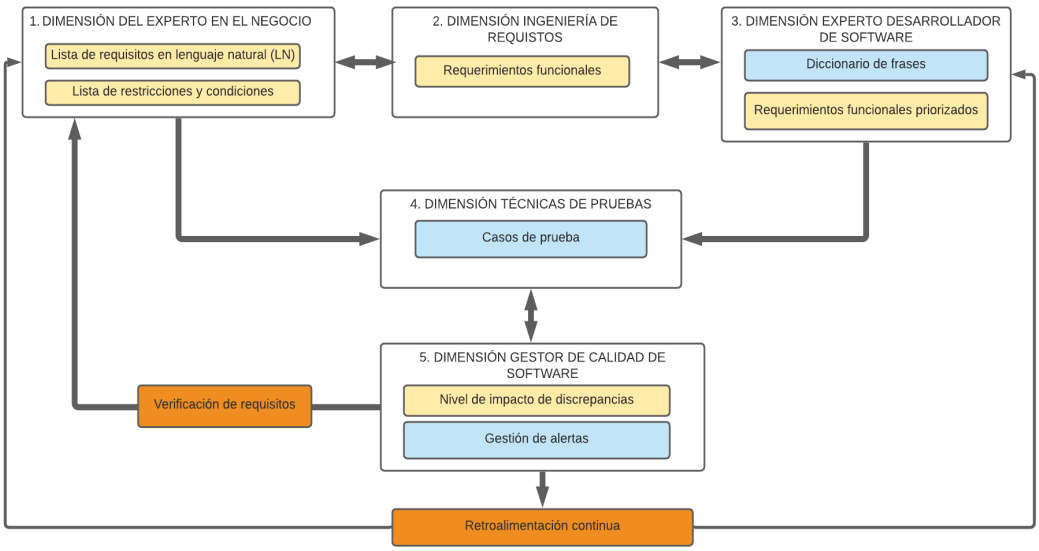


Fig. N° 5: Modelo de verificación de requisitos software, aplicado al desarrollo de software comercial.

Fuente: Elaboración propia

Dimensión Experto del negocio, permite caracterizar al stakeholder en los distintos roles que desempeña, de acuerdo a la pirámide organizacional y al rol que tendrá con respecto al software, quienes brindan los requisitos, con base a los procesos de software de los cuales se encuentran a cargo, producto de la interacción con el experto desarrollador, se genera en lenguaje natural la lista de requisitos y la lista de restricciones como también las condiciones del mismo.

Para la **dimensión de ingeniería de requisitos,** se elabora un plan de elicitación de requisitos, para ser aplicados a cada uno de los roles tipo del experto en el negocio, con el propósito de obtener la mayor cantidad de información en cuanto a los módulos de compras, ventas y almacén, organizando los mismos en requisitos funcionales y no funcionales, siendo de interés a esta investigación, precisamente los funcionales.

En la dimensión del Experto desarrollador de software, se genera la lista de los requerimientos funcionales, producto del proceso de elicitación de requisitos aplicada al experto en el negocio, organizando el **diccionario de frases** que contiene las distintas maneras en las que el experto del negocio define o describe los requisitos de su software comercial, además de la lista de requisitos priorizados.

La **dimensión técnica de pruebas** desarrollará los escenarios qué, según la naturaleza del módulo a construir, permite la elaboración de los **casos de prueba** suficientes, que permita depurar los errores y fallas por una incorrecta interpretación de requisitos.

Finalmente, **dimensión del gestor de calidad de software**, que tiene como propósito monitorear e identificar las discrepancias que se presente y a través de un mecanismo evaluar el **nivel de impacto de discrepancias,** que permita gestionar las alertas, verificadas con el experto en el negocio a través de la verificación de requisitos, aplicando una permanente retroalimentación continua.

La teoría tomó como base los estándares de calidad de software, como las ISO 12207 que en (INDECOPI, 2006) divulgar los procesos del ciclo de vida del software de la organización. Ha sido diseñado para aquellos interesados en

comprar software, así como para desarrolladores y proveedores, mostrando una amplia gama de procesos desde la recopilación de requisitos hasta la realización del software.

La referencia del estándar ISO/IEC 9126 (Abud Figueroa, Calidad de la industria del software. La norma ISO-9126, 2012) define la usabilidad como la visión del usuario de la calidad de un producto de software, cuando se usa en un entorno y contexto de uso particular. También establece factores para medir que los usuarios pueden lograr sus objetivos en un entorno particular, en lugar de medir los atributos del software en sí.

Como apoyo a la determinación en la especificación de requisitos, se consideró el estándar IEEE 1233-1998 (IEEE-SA, 1998) para el desarrollo y especificación de los requisitos del sistema, proporciona una guía para desarrollar una especificación de un conjunto de requisitos del sistema. Incluye definir, organizar, presentar y modificar requisitos. Así como el estándar IEEE 1012-1998 (IEEE C. , 2005) dónde está la verificación y validación de los procesos de software se determina, si los productos desarrollados en una operación dada cumplen con los requisitos de esa operación y si el software cumple con el uso previsto y las necesidades del usuario o no. En última instancia, eso incluye "análisis, inspección, evaluación, prueba de productos y procesos de software".

3.4 **Aporte práctico: Metodología integral de casos de prueba.**

Introducción

En este apartado se describe como una empresa desarrolladora de software aplicaría la metodología integral de casos de prueba para el desarrollo de software comercial, esta metodología surge del modelo de verificación de

requisitos, para que a través de la implementación escenarios de casos de prueba expresados en plantillas de retroalimentación continua y tomando en cuenta los estándares de calidad de software, específicamente los referidos a la verificación de requisitos, las metodologías de desarrollo de software, de tal manera que se utilizan instrumentos como herramientas, para reducir el índice de discrepancia en la interpretación de los requisitos software, entre el experto en el negocio y el experto desarrollador. Se evalúa el nivel de discrepancias en la formulación de los requerimientos funcionales, los cuales serán clasificados de manera priorizada con base la cercanía al core del negocio, evaluando el nivel de impacto en el proyecto de desarrollo de software, luego monitorear su corrección mediante la verificación y mejora de los requisitos.

Con el aporte práctico se revela la relación causa – efecto del aporte teórico; ya que se establece una correspondencia, que las retroalimenta y dinamizan mutuamente.

3.4.1. **Fundamentación de la metodología de casos de prueba para el desarrollo de software comercial.**

Iniciamos la fundamentación definiendo metodología según (Coelho, 2020) conjunto de métodos y técnicas científicas, que se aplican sistemáticamente en un proceso de investigación para obtener un resultado teóricamente válido. Además (Hernández Sampieri, Fernandez, & Baptista, 2010) define como "... un conjunto de acciones organizadas lógicamente que tienen una secuencia de trámites con el fin de alcanzar los objetivos del destino especificado". Luego, a través de casos de prueba, lo defino en (Pickin & Garcia Valls, 2013) como especificación de la interacción entre la implementación bajo prueba o el IUT (en siglas en inglés) y el software experimental, o usuarios humanos, estimular

la IUT a través de las interfaces de comunicación del software comercial con el usuario final, observando su comportamiento y reacciones. Un caso de prueba "diseñado para realizar una actuación particular o para verificar el cumplimiento de un requisito particular". Según (Villalobos Abarca, 2019) existen diferentes métodos para seleccionar el conjunto de casos de prueba:

-Método de clase de equivalencia, que divide las posibles entradas en clase para que todos los miembros sean de la misma clase de equivalencia verificaciones para la misma propiedad en el sistema.

-Otra forma es analizar los valores límite o valores límite, "Consiste en elegir como caso de prueba, los valores de entrada que se encuentran en el contorno de las clases de equivalencia (justo en un lado, justo encima del otro y justo sobre el contorno). Este método se conoce como diagrama de causa / efecto y tabla de decisión. Consiste en crear un diagrama de causa / efecto a partir de especificaciones y seleccionar suficientes casos de prueba para proporcionar una buena cobertura del gráfico, "causa" las características de los datos de entrada y afectan las clases de salida que el programa puede proporcionar. Entonces, la tabla de decisión es un defecto de base con una dependencia entre causa y efecto, para colapsar la tabla de decisión y elegir un solo caso de prueba para todas las causas que producen el efecto. Actuando igual o para cada columna de la tabla de decisiones.

Finalmente, con el método adivinación de errores, el administrador de pruebas intenta "imaginar qué error es más probable que se hayan cometido y crear casos de prueba para verificarlos".

3.4.2. **Construcción del aporte práctico**

El objetivo de la metodología integral de casos de prueba, es reducir los tiempos de especificación, como consecuencia también los excesos de tiempos de entrega del proyecto por el número de errores generados, dada la incorrecta interpretación de requisitos, entre los manifestados por el experto en el negocio al experto desarrollador de software, así como los costos por el número de requisitos no atendidos en el desarrollo del software comercial, se propone para el desarrollo del aporte práctico, seis fases:

Fase 1: Elicitación de requisitos desde el enfoque del experto en el negocio.

Fase 2: Caracterización de los requisitos desde el enfoque del desarrollador de software.

Fase 3: Análisis y priorización con base en la ingeniería de requisitos.

Fase 4: Formulación de los escenarios de casos de prueba.

Fase 5: Medición y evaluación del gestor de calidad de software.

Fase 6: Proceso de verificación de requisitos

Con una permanente retroalimentación entre cada una de las etapas, para la integración del procedimiento.

Fase 1: Elicitación de requisitos desde el enfoque del experto en el negocio, definiendo un plan de elicitación en el cual de acuerdo a la identificación los interesados, los cuales pueden ser:

- **De nivel operativo**
- **De nivel administrativo**
- **De nivel gerencial**

Una vez ubicados en un determinado nivel se caracterizan definiendo roles y responsabilidades, así como el rol de interacción con la aplicación software a desarrollar que, según la dimensión o el alcance del software comercial

determinamos, identificando los módulos solicitados, sus funciones, así como los roles que cumpliría cada interesado.

Fase 2: Caracterización de los requisitos desde el enfoque del desarrollador de software, una vez obtenidos todos los requisitos según la clasificación de (Sommerville, 2016) caracterizan en;

- **Requisitos Funcionales**
    - Del sistema
    - Del usuario
- **Requisitos no Funcionales**
    - Del producto
    - Externo
    - Organización

En la presente investigación sólo se trabajará los requisitos funcionales que se definieron las anomalías manifestadas en el desarrollo del software Comercial.

Fase 3: Análisis y priorización con base en la ingeniería de requisitos, una vez clasificados los requisitos, se organizan en la plantilla 1 de especificación de requisitos, estos deben ser priorizados de acuerdo al core del negocio y el alcance acordado del software comercial en desarrollo:

- **Alta**
- **Media**
- **Baja**

Con ello se hacen listas con los requisitos clasificados.

Fase 4: Formulación de los escenarios de casos de prueba, para lo cual se define el plantilla 2 para determinar la caracterización de los escenarios en:

- **Básico**
- **Completo**
- **De excelencia**

Estos escenarios serán seleccionados de acuerdo a la clasificación de casos de prueba según la funcionalidad evaluada.

Fase 5: Medición y evaluación del gestor de calidad de software, se establecen niveles de impacto que permita la evaluación del alcance de afectación en el proceso de desarrollo del software comercial, en niveles:

- **Alto (**Completo **–** De excelencia)
- **Medio (**Básico Completo)
- **Bajo (**Básico)

Ello nos permitirá determinar el nivel de afectación en cuanto a tiempo, recursos y economía afectada en el desarrollo del software comercial.

Fase 6: Proceso de verificación de requisitos, con los resultados de la evaluación de impacto se hace el proceso con el experto en el negocio quien complementará los mecanismos aplicados para la mejora continua de los requisitos no atendido o interpretados de manera incorrecta.

La estructura de la metodología integral de casos de prueba para el desarrollo de software comercial en la siguiente figura**:**

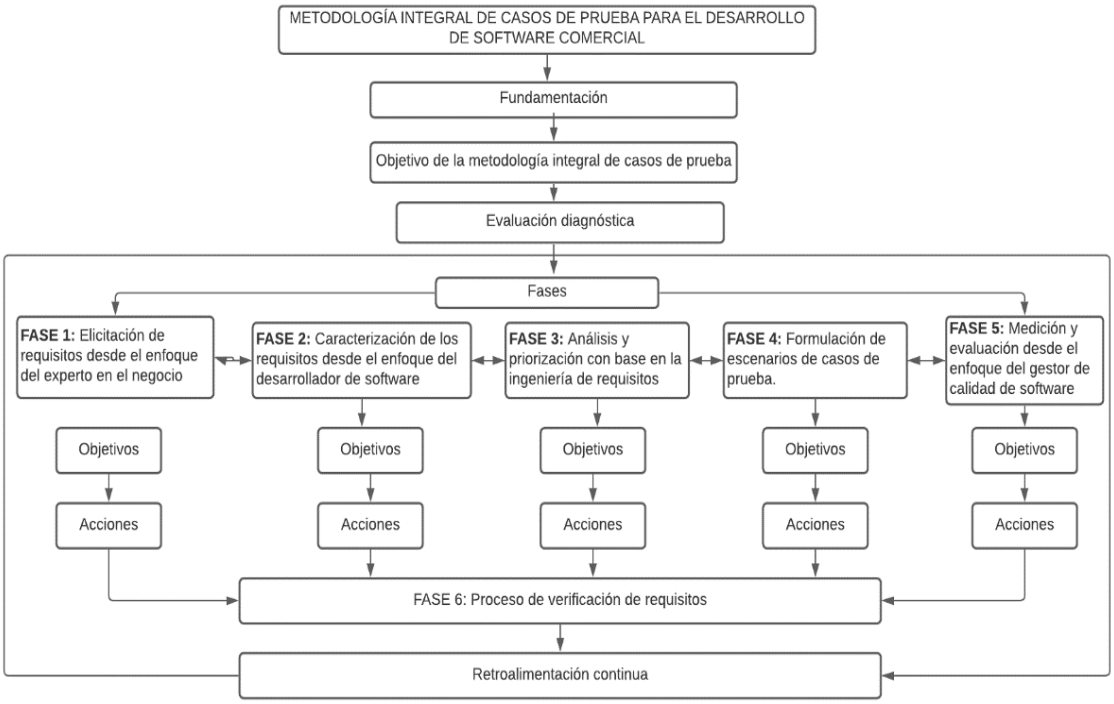


Fig. N° 6: Metodología integral de casos de prueba, para el desarrollo de software comercial.

Fuente: Elaboración propia

En una evaluación post test en una empresa desarrolladora de software de la ciudad de Chiclayo, se desarrolló:

Fase 1: Elicitación de requisitos desde el enfoque del experto en el negocio, definiendo un plan de elicitación en el cual de acuerdo a la identificación los interesados:

TIPO INTERESADO	**TÉCNICA**	**INSTRUMENTO**	**N° DE PARTICIPANTE**
De nivel operativo.	Encuesta	Cuestionario	6
De nivel administrativo.	Entrevista	Cuestionario	3
De nivel gerencial.	Entrevista	Guía de entrevista	1

Tabla N° 06: Identificación de interesados por nivel.

Fuente: Elaboración propia

Roles y responsabilidades:

EXPERTO EN EL NEGOCIO	ROL	RESPONSABILIDAD
Nivel operativo	Transaccional	Registro, movimientos, reportes.
Nivel administrativo	Ejecutivo	Gestión de la información.
Nivel gerencial.	Gerencial	Toma de decisiones en función de la gestión de la información.

Tabla N° 07: Roles y responsabilidades.

Fuente: Elaboración propia

Los módulos en los cuales interactuará el experto en el negocio serían en Compras, Ventas y Almacén, una ejemplificación:

Fase 2: Caracterización de los requisitos desde el enfoque del desarrollador de software,

REQUISITOS SOLICITADOS	**TIPO DE REQUERIMIENTO**	**SUBTIPO**
Proformas vía email	FUNCIONALES	DE SISTEMA
Disponibilidad permanente del software	FUNCIONALES	DE SISTEMA
Control de acceso al software	FUNCIONALES	DE SISTEMA
Rapidez y puntualidad en la entrega	FUNCIONALES	DE SISTEMA
Modificable a partir de una tabla de parámetros.	FUNCIONALES	DE SISTEMA
Ejecución y Reprocesos a partir de fechas.	FUNCIONALES	DE SISTEMA
Alcance del proyecto	FUNCIONALES	DE SISTEMA
Integración que pueda tener el aplicativo con otros.	FUNCIONALES	DE SISTEMA
Administración de usuarios	FUNCIONALES	DE SISTEMA
Modificación de procesos para que los clientes realicen pagos (de forma semi automática	FUNCIONALES	DE SISTEMA
Modificación de los formatos de los documentos que	FUNCIONALES	DE SISTEMA

acompañan al proceso de adquisición de productos del cliente		
Prototipos y flujo de proceso.	FUNCIONALES	DE SISTEMA
Operaciones mas rápidas y eficientes para optimizar tiempos en cargas de archivos grandes que mayormente se dan en la madrugada.	FUNCIONALES	DE SISTEMA
Automatización de procesos contables, para generar documentos de cobranza	FUNCIONALES	DE SISTEMA
Facturación electrónica	FUNCIONALES	DE SISTEMA
Como el sistema cumplirá los reglamentos y regulaciones en la FE	FUNCIONALES	DE SISTEMA
Reporte de ventas	FUNCIONALES	DE USUARIO
Control de stock y/o Kardex	FUNCIONALES	DE USUARIO
Reportes en general	FUNCIONALES	DE USUARIO
Funcionalidades del sistema y los datos que se debe manejar	FUNCIONALES	DE USUARIO
Reportes exportados al Excel o PDF	FUNCIONALES	DE USUARIO
Reporte de caja, caja diario o kardex de producto.	FUNCIONALES	DE USUARIO
Reporte de producción del personal en términos económicos, según jefe de área	FUNCIONALES	DE USUARIO
Historias de usuario claras antes del desarrollo	FUNCIONALES	DE USUARIO
Registro de ventas y compras	FUNCIONALES	DE USUARIO
Balance comercial.	FUNCIONALES	DE USUARIO
Manejo de almacén	FUNCIONALES	DE USUARIO
Kardex	FUNCIONALES	DE USUARIO
Sencilla, Correcta, Comprensible	FUNCIONALES	DE USUARIO
Que la solución represente un menor esfuerzo para el	FUNCIONALES	DE USUARIO

usuario final al realizar su trabajo		
Lo mínimo que se solicita es un documento escrito del paso a paso del proceso. Adjuntando los videos del proceso, documentos de entrada y salida, para su verificación	FUNCIONALES	DE USUARIO
Compras, Ventas, Control de Caja, Stock de artículos.	FUNCIONALES	DE USUARIO

Tabla N° 08: Clasificación y subclasificación de requisitos funcional.

Fuente: Elaboración propia

Fase 3: Análisis y priorización con base en la ingeniería de requisitos, una vez clasificados los requisitos:

Análisis

ID REQUERIMIENTOS RQ001	FECHA 10/07/2021	HORA 15:15	VERSIÓN V1	RESPONSABLE DESARROLLADOR: D1 RESPONSABLE CLIENTE R1, R2, R3
PROCESO		Ventas		
ACCIÓN		Autenticar		
OBJETO		Cliente		
DATOS A CONSIDERAR		Usuario, Contraseña, Perfil.		
RESTRICCIONES		Ninguna		
ESCENARIOS SIMILARES		Datos de usuario según módulo Usuario con privilegios por módulo		

Tabla N° 09: Plantilla de análisis: Ventas.

Fuente: Elaboración propia

ID REQUERIMIENTOS RQ002	FECHA 10/07/2021	HORA 16:00	VERSIÓN V1	RESPONSABLE DESARROLLADOR: D2 RESPONSABLE CLIENTE R3 y R4
PROCESO		Ventas		
ACCIÓN		Registrar		
OBJETO		Cliente		
DATOS A CONSIDERAR		Nombres, Apellidos, DNI, Dirección		
RESTRICCIONES		Validar DNI con el Webservice de RENIEC Categorías		
ESCENARIOS SIMILARES		Grabar datos del cliente		

	Mostrar los datos del cliente en la pantalla Reportar clientes

Tabla N° 10: Template de análisis: Venta-Registrar.

Fuente: Elaboración propia

ID REQUERIMIENTOS RQ003	FECHA 10/07/2021	HORA 10:00	VERSIÓN V1	RESPONSABLE DESARROLLADOR: D1 RESPONSABLE CLIENTE: C6
PROCESO		Facturación		
ACCIÓN		Generar una nota de crédito		
OBJETO		Nota de crédito (parcial o total)		
DATOS A CONSIDERAR		Cantidad, categoría, precio, descripción		
RESTRICCIONES		Se emite a solicitud del cliente o cuando la venta corresponda a un período o ejercicio contable distinto		
ESCENARIOS SIMILARES		Reclamos por productos comprados Devolución de productos en mal estado		

Tabla N° 11: Plantilla de análisis: Facturacìón.

Fuente: Elaboración propia

ID REQUERIMIENTOS RQ004	FECHA 12/07/2021	HORA 10:00	VERSIÓN	RESPONSABLE DESARROLLADOR: D1 RESPONSABLE CLIENTE R5
PROCESO		Compras		
ACCIÓN		Cotizar artículos		
OBJETO		Artículos		
DATOS A CONSIDERAR		Nombre, características, Descripción, unidad de medida		
RESTRICCIONES		Productos nacionales		
ESCENARIOS SIMILARES		Pedir productos agotados Buscar proveedores para la compra de artículos Comparación de criterios de compra de artículos		

Tabla N° 12: Plantilla de análisis: Facturación.

Fuente: Elaboración propia

ID REQUERIMIENTOS RQ005	FECHA 10/07/2021	HORA 10:00	VERSIÓN V1	RESPONSABLE DESARROLLADOR: D1 RESPONSABLE CLIENTE: C6
PROCESO		Almacén		
ACCIÓN		Codificar artículos		
OBJETO		Artículos		
DATOS A CONSIDERAR		ID artículo, Sategoría, Subcategoría, Descripción, Estado		
RESTRICCIONES		Productos nacionales		
ESCENARIOS SIMILARES		Organizar productos por su unidad de medida Buscar artículos por tipo Mostrar los tipos de artículos		

Tabla N° 13: Plantilla de análisis: Facturación.

Fuente: Elaboración propia

Estos deben ser priorizados de acuerdo al core del negocio y el alcance acordado del software comercial en desarrollo:

REQUISITOS SOLICITADOS	PROCESO O MÓDULO
Actualizar stock de productos	ALMACEN
Calcular impuestos	ALMACEN
Colocar código por producto	ALMACEN
Emitir boletas, facturas y guías	ALMACEN
Generar balance de almacén	ALMACEN
Implementar Kardex	ALMACEN
Implementar reportes de entrada y salida	ALMACEN
Pantalla de registro de productos	ALMACEN
Registrar proveedores	ALMACEN
Registro de código por producto	ALMACEN
Registro de productos	ALMACEN
Rotación de productos	ALMACEN
Sacar Kardex	ALMACEN
Sistema de gestión de almacén	ALMACEN
Validar cuenta del cliente	ALMACEN
Verificar Stock	ALMACEN
Cuanto tengo de mercaderia en dinero	ALMACEN
Saber si hay stock de un producto	ALMACEN
Categorizar productis	ALMACEN
Registro de fechas de vencimiento de productos	ALMACEN
Reporte de productos perdidos por daños	ALMACEN
Reportes de productos que se venden con mayor rapidez	ALMACEN

Indicar el motivo por el cual está ingresando o retirando el dinero en caja	FACTURACIÓN
Registrar los ingresos y egresos de caja.	FACTURACIÓN
Liquidación de caja diaria	FACTURACIÓN
Registrar operaciones en soles y dolares	FACTURACIÓN
Saber que gastos se han realizado	FACTURACIÓN
Saldo que hay en caja y bancos	FACTURACIÓN
Cuadrar caja	FACTURACIÓN
Cuánto pagar a la SUNAT	FACTURACIÓN
Estado de ganacias y pérdidas	FACTURACIÓN
Gastos realizados	COMPRAS
Ingresan los artículos comprados por almacen	COMPRAS
Ingresar el almacen	COMPRAS
Ingreso de compras en diferentes monedas (Soles, dólares)	COMPRAS
Ingreso de compras según condición (Crédito, contado)	COMPRAS
Registro de Compras	COMPRAS
Seleccionar el proveedor al cual se le está comprando.	COMPRAS
Reprte de compras por fechas	COMPRAS
Sumar precios de los productos comprados	FACTURACIÓN
Cargas de los débitos automáticos	FACTURACIÓN
Consolidado de reportes	FACTURACIÓN
Envío automático	FACTURACIÓN
Exportar a software contable	FACTURACIÓN
Facturación electrónica	FACTURACIÓN
Tipo de cambio	FACTURACIÓN
Vinculación de documentos	FACTURACIÓN
Desarrollar la visión general del sistema	FACTURACIÓN
Necesito de facturación electrónica.	FACTURACIÓN
Tiene procesos contables.	FACTURACIÓN
¿Es necesario tener internet?	FACTURACIÓN
¿Cuándo estará listo?	FACTURACIÓN
Actualizar plantillas	VENTAS
Búsqueda automática de clientes y presentar datos	VENTAS
Controlar las ventas	VENTAS
Correo con la factura	VENTAS
Crear la traza (archivos) de los débitos automáicos	VENTAS
Cuadre de caja diario	VENTAS
Cuadro de ventas	VENTAS
Cuanto se vendio en el dia	VENTAS
Cuanto vendi y gane gane en el mes	VENTAS
Envío de cotizaciones por correo	VENTAS
Definir formato de reportes	VENTAS
Formulario para la gestión de clientes.	VENTAS
Indicar la condición de ventas (Crédito, contado)	VENTAS
Modificación de los formatos de los documentos que acompañan al proceso de adquisición productos del cliente	VENTAS
Modificar la lógica de los datos	VENTAS
Pago a través de débitos automáticos	VENTAS
Pago al contado	VENTAS

Pantalla que muestre Stock, precio de venta	VENTAS
Que los registros generados que hayan pasado la fecha de activación se vayan a una tabla nueva.	VENTAS
Que permita contactar con un ejecutivo asignado	VENTAS
Registro de Ventas	VENTAS
Reporte de caja diaria	VENTAS
Reporte de comprobantes anulados	VENTAS
Reporte de ventas	VENTAS
Reportes de productos por vencer	VENTAS
Reproceso por fechas	VENTAS
Seleccionar la caja donde va a ingresar el dinero cuando la venta es al contado.	VENTAS
Tabla histórica de registros desactivados	VENTAS
Tabla que sorpote la cantidad de registros	VENTAS
Una Shell que saque los registros desactivados cada día.	VENTAS
Ventas al crédito y al contado	VENTAS
Vincular una venta o un contacto con un ejecutivo asignado.	VENTAS
Venta de equipos postpago	VENTAS
Venta de equipos postpago con chip	VENTAS
Venta de equipos postpago liberados	VENTAS
Venta de equipos desde la web	VENTAS

Tabla N° 14: Clasificación de requisitos funcionales

Fuente: Elaboración propia

- **Alta: FACTURACIÓN, VENTAS**
- **Media: COMPRAS**
- **Baja: ALMACEN**

Con ello se hacen listas con los requisitos clasificados.

Fase 4: Formulación de los escenarios de casos de prueba, para lo cual se define el plantilla:

ID de Caso de prueba	0001		
Prioridad	Media		
Descripción	Módulo de compras, que registra los productos para reposición.		
Módulo o proceso	Compras		
Acción	Producto no existente		
Corrección	Código autogenerado		
Elaborado por	D1	**Fecha de elaboración**	15/07/2021
Revisado (R) / Actualizado (A)	R1 A1	**Fecha de Revisión**	16/07/2021
Responsable de la prueba	R2	**Fecha de aplicación**	16/07/2021

Actividades de prueba			
No.	**Descripción Escenario**	**Resultados Esperados**	**Resultados Actuales**
1	Código de errado	Mensaje de alerta	No validado
2	Código inexistente	Mensaje de error	Validado
3	--	--	--

Conjunto de datos de Prueba			
Tipo de Dato	**Datos 1**	**Datos 2**	**Datos 3**
ALFANUMÉRICO	AB_001_A1	AB_002_A1	--
--	--	--	--
Resultados de casos de prueba		Discrepancia	

Tabla N° 15: Plantilla de escenarios

Fuente: Elaboración propia

Fase 5: Medición y evaluación del gestor de calidad de software, se establecen niveles de impacto:

- **Alto (**Completo – De excelencia)
- **Medio (**Básico Completo)
- **Bajo (**Básico)

Fase 6: Proceso de verificación de requisitos, con los resultados de la evaluación de impacto se hace el proceso con el experto en el negocio correspondiente.

3.5. **Valoración y corroboración de los resultados**

Análisis pre test y pos test.

INDICADOR	PRE TEST	POST TEST
Tiempo de especificación de requisitos funcionales por proyecto	10 días adicionales	2 días adicionales
Porcentaje de margen de error en tiempos de entrega por proyecto	40%	10%
Porcentaje de tiempos de entrega por correcciones en el proyecto	10%	2%
Porcentaje de costos adicionales en recursos humanos de ejecución por el total de recursos humanos planificados.	40%	10%
Porcentaje de costos por recursos tecnológicos utilizados por el total de recursos tecnológicos planificados	20%	5%

Tabla N° 16: Pretest – Postest

Fuente: Elaboración propia

Se concluye entonces que el proceso de aplicación de la metodología integral de casos de prueba para el desarrollo del software comercial, se logró:

- Reducir tiempos adicionales a 2 días, para validar especificaciones funcionales mal interpretados por el experto en el sistema.

- Reducir a un 10% el porcentaje de demora entre el tiempo planificado en el acuerdo de entrega del proyecto y el tiempo efectivo de culminación.

- Reducir el porcentaje de tiempo de entrega por correcciones del proyecto en un 10% para los requerimientos funcionales.

- Reducir el porcentaje a un 10% de los costos adicionales en recursos humanos de ejecución por el total de recursos humanos planificados.
- Reducir el índice porcentual a un 5% de los costos adicionales por compra de recursos tecnológicos para disminuir el retraso de los tiempos de entrega.

IV. CONCLUSIONES

- Se logró caracterizar epistemológicamente el proceso de verificación de requisitos software y su dinámica, lo que se pudo definir el modelo de verificación de requisitos del desarrollo de software comercial, como aporte teórico de la presente investigación, con 5 dimensiones y una transversal retroalimentación continua.

- Se caracterizó las tendencias históricas del proceso de verificación de requisitos software y su dinámica, en cuatro etapas evidenciando el vacío de investigación que aún persiste en la incorrecta interpretación de los requisitos software, entre los manifestados por experto en el negocio y el experto desarrollador de software, a pesar de la evolución en estrategias independientes en cada una de las fases del ciclo de vida de desarrollo de software, que justifica el punto de interés de la presente investigación.

- Se caracterizó el estado actual de la dinámica del proceso de verificación de requisitos software, como objeto de la investigación, a través de encuestas a 32 desarrolladores de software de distintas empresas a través de cuestionarios aplicados, obteniéndose los resultados de la investigación mediante un pre experimento.

- Se elaboró el modelo de la sistematización epistemológica del desarrollo del software comercial, aplicando el proceso de verificación de requisitos, plasmado en cinco dimensiones 2 caracterizando al experto en el negocio y al experto desarrollador, 2 dimensiones que caracterizan los enfoques de la Ingeniería de requisitos como la de la ingeniería de requisitos y la dimensión de calidad del software.

- Se formuló la metodología integral de casos de prueba de 6 fases, concretizada a partir del aporte teórico aplicando las actividades en cada una de las fases y complementándose con instrumentación de plantillas de retroalimentación continua, que permitan la evaluación con indicadores que determinan el impacto que generan las discrepancias, que deben ser mitigadas interactuando con el experto en el negocio para mitigarlas, con base en un modelo de verificación de requisitos software, aplicado al desarrollo de software comercial.

- Se logró validar corroborando los resultados de la investigación mediante un pre experimento, en una empresa desarrolladora de software de la ciudad de Chiclayo y con la aplicación de la metodología integral de casos de prueba se pudo corroborar, midiendo indicadores de tiempos de entrega que se reduce en 30%, el indicador de excedentes económicos en recursos humanos y tecnológicos en un promedio de 30%, por correcciones de fallos una reducción del 8% lo cual mejora el desarrollo de sus software comercial.

V. RECOMENDACIONES

-Aplicar la metodología integral en las diferentes empresas de desarrollo de software, no sólo en el ámbito comercial, sino también el sector de servicios.

-Sensibilizar al experto del negocio acerca de su valiosa participación en el proceso de verificación de requisitos.

- Presentar la propuesta a la institución APESOFT (Asociación peruana de desarrolladores de software y servicios relacionados.

- Implementar la propuesta tecnológica con herramientas de inteligencia artificial, para habilitar los mecanismos propuestos de mejora continua en el proceso de verificación de requisitos.

VI. REFERENCIAS

Abud Figueroa, M. (30 de Enero de 2012). *Calidad de la industria del software. La norma ISO-9126*. Obtenido de http://148.204.210.204/revistaupiicsa/34/34-2.pdf: http://148.204.210.204/revistaupiicsa/34/34-2.pdf

Abud Figueroa, M. (30 de 01 de 2012). *Calidad del Software*. Obtenido de Calidad del Software: http://148.204.210.204/revistaupiicsa/34/34-2.pdf

ACM. (5 de Mayo de 2020). *Association for computing Machinery*. Obtenido de Association for computing Machinery: https://www.acm.org/about-acm/about-the-acm-organization

Anton, G., Earp, J., & Alspaugh, T. (2001). The Role of Policy an Stakeholder Privacy Values in Requirements Engineering. *Proceedings of Fifth IEEE International Symposium on Requirements Engineering*, 138-145.

Arango, G., Schafer, W., & Prieto, R. (1993). *Domain Analysis Methods – Software Reusability*. Ellis Horwood Ltd.

Bayona-Oré, S., Chamilco, J., & Perez, D. (2019). Mejora de procesos software: Gestión de requisitos, verificación y validación. *14th Iberian Conference on Information Systems and Technologies (CISTI)*, (pág. 6). Coimbra.

Beizer, B. (1990). *Software Testing Techniques.* Van Nostrand.

Benner, K., Feather, M., & Johnson, L. (1993). Utilizing Scenarios in the software development Process. *Proceedings of the 8th. Knowledge-Based Software Engineering Conference (KBSE 93.* IEEE.

Bird, D., & Munoz, C. (1983). Automatic generation of random self-checking test. *IEEE Xplore*, 229-245.

Boehm, B. (2006). Una visión de la ingeniería de software de los siglos XX y XXI. (págs. 12-29). Los Ángeles: University of Southern California,.

Boyer., R., Elpas., B., & Levitt., K. (1975). A formal system for testing and debugging programs by symbolic execution. *". Not. 10, 6 tJune I*, 234-245.

Canfora, G. (2004). *Software Evolution in the Era of Software Services.* Benevento: IEEE Computer Society.

Carrizo, D., & Alfaro, A. (2018). Método de aseguramiento de la calidad en una metodología. *Ingeniare. Revista chilena de ingeniería*, 114-129.

Carroll, J. (1995). Scenario-Based Design: Envisioning Work and Technology in System Development. En J. W. Sons, *Introduction: The Scenario Perspective on System Developmen.* New York: J. Carroll, ed.

Chao, Y., Qing, L., Kiu, L., Yuwen, C., & Hailong, W. (2021). Industrial Design and developmente software system architecture based on model-based system engineering and cloud computing. *Elsevier*, 401-423.

Clarke, L. (1976). A system to generate test data and symbolically execute programs. *IEEE*, 215-222.

Coelho, F. (13 de Agosto de 2020). *Significados*. Obtenido de Significados: https://www.significados.com/metodologia/

Deutsch, M. (1979). *Verification and validation, Software Engineering.* Prentice Hall.

Díaz, R. (2002). *Reutilización de requisitos funcionales de sistemas distribuidos utilizando técnicas de descripción formal. Tesis doctoral.* España: Dialnet.

Dorfman, M., & Thayer, R. (1997). *Software Engineering.* IEEE Compouter Society Press.

GeeksforGeeks. (1 de Marzo de 2021). Obtenido de https://www.geeksforgeeks.org/software-testing-techniques/

George, C., & Haxthausen, A. (2003). The logic of the RAISE specification language. *Computing and Informatics*, 323-350.

Gough, P., A., ,Fodemski, F. , T., Higgins, S., A., & Ray, S., J. (1995). Scenarios - An Industrial Case Study and Hypermedia Enhancements. *RE95: Proceedings of the International Symposium on Requirements Engineering, IEEE Computer Society Press*, (págs. 10-17). Los Alamitos, California.

Guezzi, C., Jazayeri, M., & Mandrioli, D. (1991). *Fundamentals of Software Engineering.* México: Prentice Hall.

Gutierrez R, J. (2011). *Generación de pruebas del Sistema a partir de la especificación funcional.* Sevilla: Universidad de Sevilla.

Hadad, G., Doorn, J., Kaplan, G., & Leite, J. (1999). Enfoque MiddleOut en la Construcción e Integración de Escenarios. *Enfoque MiddleOut en la Construcción e Integración de Escenarios". Proceedings de WER´99*, (págs. 79-94). Buenos Aires, Argentina.

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación.* México: Mc Graw Hill.

Hernández Sampieri, R., Fernandez, C., & Baptista, P. (2010). *Metodología de la investigación. (5ª ed.).* Chile: McGraw-Hill.

Hetzel, W., & Hetzel, B. (1988). The complete guide to software testing. *QED Information Sciences*.

Hoare, C. (1994). Programming: Sorcery or Science? *IEEE Software*, 33-41.

Howden, W. (1982). Life-Cycle Software Validation. *IEEE Computer*, 71-78.

Hu, X., Zhuang, Y., & Zhang, F. (2019). A security modeling and verification method of embedded software based on Z and MARTE. *Elsevier*, 22.

IEEE. (1990). *IEEE standard glosasry of software engineering terminology.* New York: IEEE Computer Society.

IEEE. (1991). 829-1983 - IEEE Standard for Software Test Documentation. *IEEE Xplore*.

IEEE. (2009). 1008-1987 - IEEE Standard for Software Unit Testing. *IEEE Xplore*.

IEEE, C. (2005). *IEEE Standad for software verification y validation.* New York: The Insitute of electru'ical and electronics engineers. Inc.

IEEE-SA, S. (1998). *IEEE guide for developing system requirements specifications.* New York: The institute of Electrical and electronics Engineers, Inc.

INDECOPI. (2006). *Norma Técnica peruana ISO 12207.* Lima.

ISO, N. (2001). *ISO / IEC 14598-6: 2001*. Ginebra: ISO / IEC 2001 International estándar.

ISO25000. (26 de Junio de 2018). *Calidad de Software y datos*. Obtenido de https://iso25000.com/index.php/normas-iso-25000

Korel, B. (1990). Automated test data generation. *IEEE Transaction Software*, 870-879.

Kotonya, G., & Somerville, I. (1998). *Requirements Engineering, Processes and Techniquess.* 1998: John Wiley & Sons.

Leite, J. (1997). *Ingeniería de Requisitos.* Notas de Cátedra.

Myers, G., Badgett, T., & Sandler, C. (2012). *The art of software testing.* Canadá: John Wiley & Sons, Inc., Hoboken, New Jersey.

Nauer, P., & Randall, B. (1969). *Software Engineering, NATO.* Brussels: Belgium.

Normas ISO. (10 de Noviembre de 2018). *Normas ISO*. Obtenido de Normas ISO: https://www.normas-iso.com/iso-iec-15504-spice/

Organización Internacional de Normalización. (12 de Octubre de 2020). *ISO*. Obtenido de https://www.iso.org/about-us.html

Özakıncı, R., & Tarhan, A. (2018). Early Software Defect Prediction: A Systematic Map and Review. *The Journal of Systems & Software*, 1-40.

Pauta Ayabaca, L., & Moscoso Bernal, S. (2017). Verificación y Validación de Software. *Kilkana Técnica*, 25-32.

Pickin, S., & Garcia Valls, M. (2013). *Introducción a la Ingeniería de software.* Madrid: Universidad Carlos III.

Pohl, K., & Metzger, A. (2006). *Variability Management in Software Product Line.* Shangai: ACM.

Pradel Miquel, J., & Raya Martos, J. (1981). *Introducción a la ingeniería del software.* Catalunya: UOC.

Pressman, R. (2015). *Ingeniería de software. Un enfoque práctico.* México: Mc Graw Hill.

Produce. (5 de Febrero de 2016). *Ministerio de la producción.* Obtenido de http://ogeiee.produce.gob.pe/index.php/shortcode/oee-documentos-publicaciones/publicaciones-anuales/item/758-las-mipyme-en-cifras-2016

PromPerú. (2020). Perú soluciones y servicios que traspasan fronteras, los 100 proveedores de servicio más innovadores del país. *Perú xpert*, 88.

Ramamoorthy, C. (1976). On the automated generation of program test data. *IEEE transaction software*, 293-300.

Ramamoorthy, C., Ho, S., & Chen, W. (1976). On the Automated Generation of Program Test Data. *IEEE*, 293-300.

Ramírez Ramos, A. (2018). ISO/IEC 25000: System and Software Quality Requirements and Evaluation. *Revista SG*.

Royce, W. (1991). *Current Problems in "Aerospace Software Engineering: A Collection of Concepts"*. Washintong: American Institute of Aeronautics, Inc.

Sánchez Peño, J. (2015). *Pruebas de software. Fundamentos y técnicas.* Madrid: Universidad Politécnica de Madrid.

Somerville, I. (2011). *Ingeniería de Software.* México: Pearson.

Sommerville, I. (2016). *Software Enginneering.* Scotland: Pearson.

SPR. (s.f.). Aframework for adopting software Process. *Springer*.

Thayer, R., Dorfman, M., & Foreword by Davis, A. (1997). Software Requirements Engineering. *IEEE Computer Society Press*, 30.

Toval Álvarez, J. (2009). *Una propuesta de gestión integrada de modelos y requisitos en líneas de productos software.* Murcia: Universidad de Murcia.

Turing, A. (1950). Computing machinery and inteligence. *Mind*, 433-460.

Villalobos Abarca, M. (2019). *Proyectos de desarrollo de software.* Arica: Universidad de Tarapacá.

Von Bertalanffy, L. (1976). *Teoría general de los sistemas.* New York: Fondo de cultura económica.

Wang, Y., & Wang, Y. (2008). *Software Process in Software Project Management.* IEEE Computer Society.

Watts, H. (1989). *Managing the software process.* Addison-Wesley.

Xavier, E. S. (2015). *Calidad del software: técnicas de prevención, detección y corrección de defectos.* Cataluña: Universitat Oberta de Catalunya.

Zamora, J. (2011). *Análisis de los procesos de verificación y validación en las organizaciones software.* Madrid: Repositorio institucional de la Universidad Carlos III.

Zhang, B., & Wang, C. (2011). Automatic generation of test data for path testing by adaptive genetic simulated annealing algorithm. *IEEE*, 38-42.

Zhang, H., Kitchenham, B., & Jeffery, R. (2007). A Framework for Adopting Software Process Simulation in CMMI Organizations. *Springer*, 320-331.

Zhang, J., & Li, J. (2020). Testing and verification of neural-network-based safety-critical control software: A systematic literature review. *Elsevier*, 21.

ANEXOS

ANEXO N° 1 MATRIZ DE CONSISTENCIA

Manifestaciones del problema	- Se identifica más de una interpretación de los requisitos por parte del experto desarrollador del software, frente a los planteados por el experto en el negocio. - Se emplea tiempo y mayores costos, en la elaboración de una aplicación software, con especificaciones de requisitos errados. - Los costos de corrección de los errores detectados son más elevados, cuando son identificados, en las últimas etapas del desarrollo de software. - Altos índices de modificación de requisitos, durante la implementación de la aplicación software. - Se requiere de recurso humano adicional para cubrir los desfases en tiempo del desarrollo de la aplicación software - Se hace uso de recursos hardware, sobre utilizados, cuando se elabora la aplicación software. - Se define un nivel de insatisfacción del experto en el negocio, que requiere la aplicación software, con respecto al producto entregado por el experto desarrollador. - El proceso para el desarrollo del proyecto, aplica técnicas de pruebas de software con base en su experiencia, en alguna

	etapa del desarrollo de software. - La estructura organizacional, de las empresas desarrolladoras de aplicaciones, no es clara en cuanto a los roles y responsabilidades. - Escasa participación del experto en el negocio, que requiere la aplicación software, en el proceso de verificación de los requisitos funcionales, en el ciclo de vida de desarrollo. - Se generan problemas de aceptación entre los expertos del negocio que requieren la aplicación software y lo entregado por los expertos desarrolladores. - El jefe de proyecto desestima la importancia del impacto de truncar, la actividad de verificación de requisitos. - Se carece de un conjunto de acciones que de manera integral apliquen pruebas de aceptación de software, para que a través de un proceso de verificación se diagnóstique si están implementando los requisitos solicitados.
Problema	Insuficiencias en el proceso de verificación de los requisitos software, limitan del desarrollo de software comercial, según los instrumentos aplicados, se identifican las siguientes causas
Causas que originan el Problema	- La insuficiente integración de los casos de pruebas, en la aplicación del proceso de verificación de requisitos, limita hacer seguimiento del desarrollo. - Insuficiente referencia teórica sobre cómo se aplican los casos de prueba en el desarrollo de software, de forma integral, como proceso de verificación de requisitos.

	- Escaso uso de modelos formales, en las empresas desarrolladoras, que determinen una correcta interpretación por parte del experto desarrollador, con respecto a los requisitos software planteados por el experto en el negocio, debido a que no se aplica un correcto proceso de verificación de los requisitos. - Los métodos de verificación basados en casos de prueba en su mayoría son estáticos y además, no contemplan un enfoque integral, con procesos de retroalimentación continua, en la verificación de los requisitos durante las etapas del desarrollo software. - Las organizaciones practican una cultura de desarrollo de aplicaciones software de modo empírico, basada en su experiencia y en sus habilidades prácticas.
Objeto de la Investigación	Proceso de verificación de los requisitos de software.
Objetivo General de la Investigación	Aplicar una metodología integral de casos de prueba, sustentado en un modelo de verificación de requisitos software, con un enfoque sistémico aplicando procesos de retroalimentación continua, para desarrollo de software comercial.
Objetivos específicos	- Caracterizar epistemológicamente el proceso de verificación de requisitos software y su dinámica. - Caracterizar las tendencias históricas del proceso de verificación de requisitos software y su dinámica. - Caracterizar el estado actual de la dinámica del proceso de verificación de requisitos software, en una empresa desarrolladora de Chiclayo. - Elaborar el modelo de la sistematización epistemológica del

	desarrollo del software comercial. - Elaborar una metodología integral de casos de prueba a través de plantillas de retroalimentación continua, basado en un modelo de verificación de requisitos software, aplicado al seguimiento de desarrollo de software comercial. - Validar corroborando los resultados de la investigación mediante un pre experimento.
Campo de la investigación.	Dinámica del proceso de verificación de requisitos.
Título de la Investigación.	Metodología integral de casos de prueba sustentado en un modelo de verificación de requisitos para desarrollo de software comercial.
Hipótesis	Si se aplica una metodología integral de casos de prueba a través de plantillas de retroalimentación continua, con base en un modelo de verificación de requisitos software, que tenga en cuenta la relación de la integración de métodos de caso de prueba y las etapas de elaboración, entonces se contribuye al desarrollo de software comercial.
Variables	**Variable independiente:** - Metodología integral de casos de prueba a través de plantillas de retroalimentación continua, basado en un modelo de verificación de requisitos software. **Variable dependiente:** -Desarrollo de software comercial.

Anexo 02: Operacionalización de las variables.

Variable dependiente	Dimensión	Indicador	Definición	Técnicas e Instrumentos	Fuente de verificación
Desarrollo de software comercial.	**Orientada al experto en software:** Es quien en base a los requerimientos funcionales del software definidos por el experto en el negocio desarrolla la aplicación.	Tiempo de especificación de requisitos funcionales por proyecto	Indica los tiempos que se adicionan para validar especificaciones funcionales mal interpretados por el experto en el sistema.	- Entrevistas - Observación - Herramientas de monitoreo - Análisis documental	Personal de TI Fichas técnicas Reportes de monitoreo
		Margen de error en tiempos de entrega por proyecto	Es la demora entre el tiempo planificado en el acuerdo de entrega del proyecto y el tiempo efectivo de culminación.		
		Porcentaje de tiempos de entrega por correcciones en el proyecto	Fluctuación del porcentaje de tiempo adicional que se extiende el proyecto por correcciones de requerimientos funcionales mal interpretados.		
		Porcentaje de costos de recursos humanos de ejecución por recursos humanos planificados	Porcentaje de costos adicionales por contratación de recursos humanos adicionales para disminuir el retraso de los tiempos de entrega.		
		Costos de recursos tecnológicos utilizados por recursos tecnológicos planificados	Costos adicionales por compra de recursos tecnológicos adicionales para disminuir el retraso de los tiempos de entrega.		

	Orientada al experto en el negocio: es quien establece los requerimientos funcionales del software	Margen de error en los tiempos de entrega.	Nivel de discrepancia entre el tiempo planificado de entrega y tiempo de entrega efectivo del proyecto.	- Encuestas - Entrevistas - Análisis documental	Usuarios Personal de TI Fichas técnicas
		Costos por requerimientos funcionales adicionales.	Valorización de costos adicionales por acuerdos no establecidos debido a la interpretación equivoca del alcance de los requerimientos funcionales.		
		Satisfacción del usuario	Grado de satisfacción al asegurar la calidad de experiencia del usuario por acuerdos de servicios, en el desarrollo del proyecto.		

VARIABLES	DIMENSIONES	DESCRIIPCIÓN
V. INDEPENDIENTE **Metodología integral de casos de prueba sustentado en un modelo de verificación de requisitos**	**Introducción-Fundamentación**.	Se establece el contexto y ubicación de la problemática a resolver. Ideas y puntos de partida que fundamentan la estrategia. Se indica la teoría en que se fundamenta el aporte propuesto.
	II. Diagnóstico-	Indica el estado real del objeto y evidencia el problema en torno al cual gira y se desarrolla la estrategia, protocolo, o programa, según el aporte practico a desarrollar.
	Planteamiento del objetivo general.	Se desarrolla el objetivo general del aporte práctico. Se debe tener en cuenta que no es el de la investigación.
	Planeación estratégica	- Se definen metas u objetivos a corto y mediano plazo que permiten la transformación del objeto desde su estado real hasta el estado deseado. Planificación por etapas de las acciones, recursos, medios y métodos que corresponden a estos objetivos. -Se debe tener en cuenta las dimensiones de la operacionalización de la variable dependiente.
	Instrumentación	Explicar cómo se aplicará, bajo qué condiciones, durante qué tiempo, responsables, participantes.

	Evaluación	Definición de los logros obstáculos que se han ido venciendo, valoración de la aproximación lograda al estado deseado.

USS | ESCUELA DE POSGRADO

ANEXO N° 3 INSTRUMENTO

Encuesta al Desarrollador

Estimado (a) desarrollador:
Con la presente encuesta se pretende analizar el estado actual del proceso de obtención de los requisitos, que limita el desarrollo de software comercial. Por ello, se solicita por favor responder con toda claridad a las siguientes preguntas. De antemano agradezco su valioso aporte que tiene carácter anónimo.

Dimensión: Calidad en la verificación de Requisitos

1. ¿Describa el procedimiento que usted realiza para la toma de requisitos en el desarrollo de un software comercial?

2. ¿Cuáles son los requisitos que con más frecuencia son solicitados por los usuarios para el desarrollo de un software comercial?

3. En base a cada uno de los procesos a implementar en una aplicación de software comercial, ¿cuáles son los términos utilizados por los usuarios al describir sus requerimientos? (Señalar lo que el usuario pide en su terminología)

Nombre del proceso	**Actividades solicitadas por el usuario**

Nombre del proceso	**Actividades solicitadas por el usuario**

Nombre del proceso	**Actividades solicitadas por el usuario**

Nombre del proceso	**Actividades solicitadas por el usuario**

Dimensión: Ciclo de vida de desarrollo de software

4. ¿En el proceso de desarrollo de la aplicación software tienen una metodología específica que consideran o que mecanismos aplican?

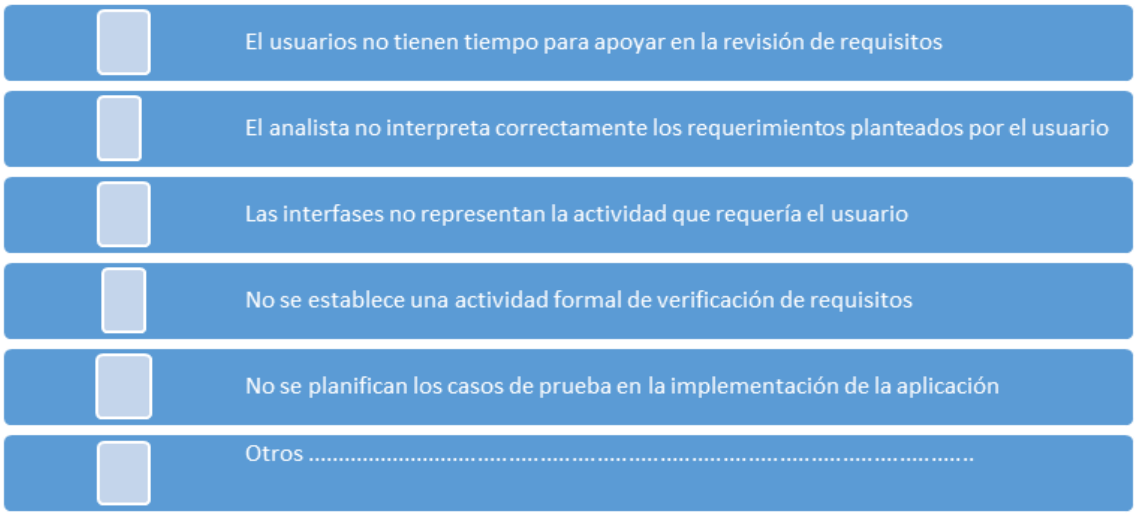
5. Mencione cuáles son los principales procesos a implementar en el desarrollo de un software comercial.

Dimensión: Gestión del seguimiento del software

6. ¿Con qué frecuencia los requerimientos del usuario no son interpretados por el desarrollador de la misma manera y que incidencias genera?

7. ¿Cuáles son los escenarios más frecuentes de falta coincidencias entre los requisitos solicitados y los implementados?

8. Mencione las incidencias que se manifiestan para dar pase a cada una de las etapas (Análisis, Diseño, implementación y pruebas) de desarrollo de software comercial.



Gracias por su apoyo

ESCUELA DE POSGRADO

ANEXO N° 4 INSTRUMENTO DE VALIDACION NO EXPERIMENTAL POR JUICIO DE EXPERTOS

INSTRUMENTO DE VALIDACIÓN NO EXPERIMENTAL POR JUICIO DE EXPERTOS

1. NOMBRE DEL JUEZ		BRAVO JAICO JESSIE LEILA
2.	PROFESIÓN	INGENIERO DE COMPUTACIÓN Y SISTEMAS
	ESPECIALIDAD	Seguridad de la información y Transformación digital
	GRADO ACADÉMICO	DOCTOR
	EXPERIENCIA PROFESIONAL (AÑOS)	27 años
	CARGO	Docente Universitaria
Título de la Investigación: METODOLOGÍA INTEGRAL DE CASOS DE PRUEBA SUSTENTADO EN UN MODELODE VERIFICACIÓN DE REQUISITOS PARA DESARROLLO DE SOFTWARE COMERCIAL.		
3. DATOS DEL TESISTA		
3.1	NOMBRES Y APELLIDOS	ARANGURÍ GARCÍA MARÍA YSABEL
3.2	PROGRAMA DE POSTGRADO	DOCTORADO EN CIENCIAS DE LA COMPUTACION Y SISTEMAS.
4. INSTRUMENTO EVALUADO		1. Entrevista ( ) 2. Cuestionario (X) 3. Lista de Cotejo ( ) 4. Diario de campo ( )
5. OBJETIVOS DEL INSTRUMENTO		<u>GENERAL</u> Analizar el estado actual del proceso de obtención de los requisitos, que limita el desarrollo de software comercial

	ESPECÍFICOS - Recoger información de los desarrolladores de software con fines académicos, excluyendo datos confidenciales - Procesar los datos de las características de la elicitación de requisitos.

A continuación, se le presentan los indicadores en forma de preguntas o propuestas para que Ud. los evalúe marcando con un aspa (x) en "A" si está de ACUERDO o en "D" si está en DESACUERDO, SI ESTÁ EN DESACUERDO POR FAVOR ESPECIFIQUE SUS SUGERENCIAS

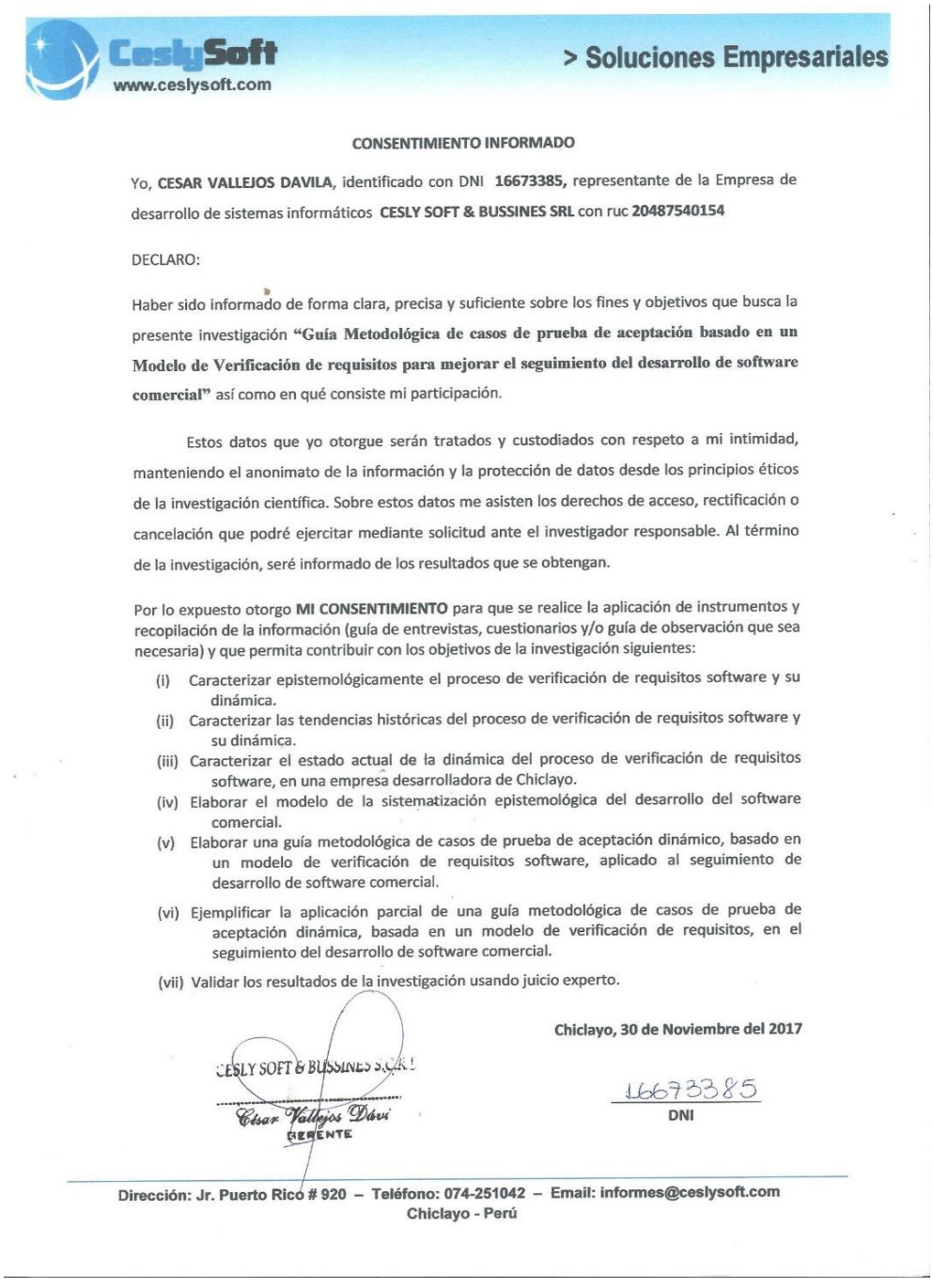
N	DETALLE DE LOS ITEMS DEL INSTRUMENTO	
01	Pregunta del instrumento ¿Describa el procedimiento que usted realiza para la toma de requisitos en el desarrollo de un software comercial? Escala de medición	A(x) D ( ) SUGERENCIAS:
02	Pregunta del instrumento ¿Cuáles son los requisitos que con más frecuencia son solicitados por los usuarios para el desarrollo de un software comercial? Escala de medición	A(x) D ( ) SUGERENCIAS:
03	Pregunta del instrumento En base a cada uno de los procesos a implementar en una aplicación de software comercial, ¿cuáles son los términos utilizados por los usuarios al describir sus requerimientos? (Señalar lo que el usuario pide en su terminología) Escala de medición	A(x) D ( ) SUGERENCIAS:
04	Pregunta del instrumento ¿En el proceso de desarrollo de la aplicación software tienen una metodología específica que consideran o que mecanismos aplican?	A(x) D ( ) SUGERENCIAS:

	Escala de medición		
05	Pregunta del instrumento Mencione cuáles son los principales procesos a implementar en el desarrollo de un software comercial. Escala de medición	A(x)	D ( )
		SUGERENCIAS:	
06	Pregunta del instrumento ¿Con qué frecuencia los requerimientos del usuario no son interpretados por el desarrollador de la misma manera y que incidencias genera? Escala de medición	A(x)	D ( )
		SUGERENCIAS:	
07	Pregunta del instrumento ¿Cuáles son los escenarios más frecuentes de falta coincidencias entre los requisitos solicitados y los implementados? Escala de medición	A(x)	D ( )
		SUGERENCIAS:	
08	Pregunta del instrumento Mencione las incidencias que se manifiestan para dar pase a cada una de las etapas (Análisis, Diseño, implementación y pruebas) de desarrollo de software comercial. Escala de medición	A(x)	D ( )
		SUGERENCIAS:	
	PROMEDIO OBTENIDO:	A(x)	D ( ):
6	COMENTARIOS GENERALES		
7	OBSERVACIONES		

Juez Experto

Colegiatura Nº 71194

Anexo 05: Consentimiento Informado



CeslySoft
www.ceslysoft.com
> Soluciones Empresariales

CONSENTIMIENTO INFORMADO

Yo, **CESAR VALLEJOS DAVILA**, identificado con DNI **16673385,** representante de la Empresa de desarrollo de sistemas informáticos **CESLY SOFT & BUSSINES SRL** con ruc **20487540154**

DECLARO:

Haber sido informado de forma clara, precisa y suficiente sobre los fines y objetivos que busca la presente investigación **"Guía Metodológica de casos de prueba de aceptación basado en un Modelo de Verificación de requisitos para mejorar el seguimiento del desarrollo de software comercial"** así como en qué consiste mi participación.

Estos datos que yo otorgue serán tratados y custodiados con respeto a mi intimidad, manteniendo el anonimato de la información y la protección de datos desde los principios éticos de la investigación científica. Sobre estos datos me asisten los derechos de acceso, rectificación o cancelación que podré ejercitar mediante solicitud ante el investigador responsable. Al término de la investigación, seré informado de los resultados que se obtengan.

Por lo expuesto otorgo **MI CONSENTIMIENTO** para que se realice la aplicación de instrumentos y recopilación de la información (guía de entrevistas, cuestionarios y/o guía de observación que sea necesaria) y que permita contribuir con los objetivos de la investigación siguientes:

(i) Caracterizar epistemológicamente el proceso de verificación de requisitos software y su dinámica.
(ii) Caracterizar las tendencias históricas del proceso de verificación de requisitos software y su dinámica.
(iii) Caracterizar el estado actual de la dinámica del proceso de verificación de requisitos software, en una empresa desarrolladora de Chiclayo.
(iv) Elaborar el modelo de la sistematización epistemológica del desarrollo del software comercial.
(v) Elaborar una guía metodológica de casos de prueba de aceptación dinámico, basado en un modelo de verificación de requisitos software, aplicado al seguimiento de desarrollo de software comercial.
(vi) Ejemplificar la aplicación parcial de una guía metodológica de casos de prueba de aceptación dinámica, basada en un modelo de verificación de requisitos, en el seguimiento del desarrollo de software comercial.
(vii) Validar los resultados de la investigación usando juicio experto.

Chiclayo, 30 de Noviembre del 2017

CESLY SOFT & BUSSINES S.C.R.L.
César Vallejos Dávi
GERENTE

16673385
DNI

Dirección: Jr. Puerto Rico # 920 – Teléfono: 074-251042 – Email: informes@ceslysoft.com
Chiclayo - Perú

ANEXOS N° 6 APROBACIÓN DEL INFORME DE TESIS

El Docente: Dr Juan Carlos Callejas Torres

De la Asignatura:

SEMINARIO DE INVESTIGACIÓN VI: INFORME DE TESIS

APRUEBA:

El Informe de Tesis: "METODOLOGÍA INTEGRAL DE CASOS DE PRUEBA SUSTENTADO EN UN MODELO DE VERIFICACIÓN DE REQUISITOS PARA DESARROLLO DE SOFTWARE COMERCIAL"

Presentado por:
Arangurí García María Ysabel

Chiclayo,19 de diciembre del 2021

_____
Dr. Callejas Torres Juan Carlos