compAS
Grupo de capacitación e investigación pedagógica

Modelo de verificación de requisitos para software comercial en el contexto peruano

María Ysabel Arangurí García
Jessie Leila Bravo Jaico
Juan Carlos Callejas Torres
Mariana Chavarry Chankay
Fiorella Vanessa Li Vega

Modelo de verificación de requisitos para software comercial en el contexto peruano

compAS
Grupo de capacitación e investigación pedagógica

Autor
María Ysabel Arangurí García
Jessie Leila Bravo Jaico
Juan Carlos Callejas Torres
Mariana Chavarry Chankay
Fiorella Vanessa Li Vega

Título del libro
Modelo de verificación de requisitos para software comercial en el contexto peruano

ISBN: 978-9942-33-621-7

Publicado 2022 por acuerdo con los autores.

Guayaquil-Ecuador

@grupocompas.ec
compasacademico@icloud.com

compÁs
Grupo de capacitación e investigación pedagógica

TABLA DE CONTENIDOS

INTRODUCCIÓN

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 del desarrollo de software, con la finalidad de garantizar un alto nivel de eficiencia en la eliminación de defectos, con costos y plazos reducidos («Ingeniería del software. Un enfoque práctico, 7ma Edición – Roger S. Pressman», 2020).

Según ACM Association for Computing Machinery (*About ACM Resources*, s. f.) la ciencias de la computación en la línea de investigación de la Ingeniería de Software, plantea parámetros de calidad para medir los aspectos de su proceso de desarrollo y garantizar procedimientos que reduzca los defectos o errores en la producción de software.

Según (*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.), 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, así como también el 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 del modelo 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 & Rojas, 2018)

(*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.) 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 (Sabadell, s. f.) 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, es el seguimiento de requisitos específicos, tanto funcionales y no funcionales. Así mismo, para los requisitos en (*NORMAS ISO 25000*, s. f.) se 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. (Ayabaca & Moscoso Bernal, 2018) Según el Informe, (*LAS MIPYME EN CIFRAS 2016*, s. f.) 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) (*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.), 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, es decir de acuerdo a 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 (Souza, s. f.), 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 (*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.) 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, quienes están legalmente autorizados. Aun así, la digitalización del consumo, según el informe Perú Service Summit 2020 (*LAS MIPYME EN CIFRAS 2020*, s. f.) 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.

Para los argumentos antes expuestos y para dar soporte a un desarrollo con calidad de la aplicación software, se propone un modelo de verificación de requisitos software, con un enfoque sistémico, con procesos de retroalimentación continua, para la mejora de procedimientos en el desarrollo de software comercial, que integre con hitos de entrada/salida, los casos de prueba, mejorando el seguimiento con un punto de vista integral para la aplicación software. (Yamada et al., 2019) las empresas en el contexto peruano, no son ajenas 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 de software detectados, son más elevados que cuando son identificados, en las últimas etapas del ciclo de vida 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** de 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 de 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 no contemplan un enfoque integral, con procesos de retroalimentación continua, en la verificación de los requisitos durante las etapas del desarrollo de software, haciéndolos de manera aislada en cada una de las mismas.
- Las organizaciones practican una cultura de desarrollo de aplicaciones de 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:

Aplicado con la intención de comprobar si es que el sistema cumple con los requerimientos específicos (*Ingenieria De Software Pressman 7 Edicion Pdf*, s. f.) pero llevándose a cabo en cada una de las etapas con diferentes técnicas y no de manera integral, sin una retroalimentación continua que genere un seguimiento exhaustivo desde la primera etapa hasta la puesta en marcha del producto software.

Según (Yamada et al., 2019), 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 de una garantía de la calidad.

Según (Redondo, 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, es la implementación del algoritmo de verificación, muchas veces con base en técnicas de modelado de control, por lo que 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 alto nivel de abstracción, como los requisitos funcionales y la información de verificación, en comparación con los elementos de abstracción de bajo nivel, como el código fuente, que, además de parecer simple, es menos rentable y atractivo.

(Mohammed et al., 2020) analiza los beneficios de adoptar un proceso de prueba en el desarrollo de software, formalmente conocido como "verificación y validación", que según él afecta no sólo el desarrollo inmediato del proyecto, sino que también se extiende al resto del negocio, para diferentes campos de actividad e incluso con clientes finales. Considera la importancia de lograr "un mayor nivel de eficiencia en el desarrollo de software" porque requiere un enfoque holístico que, aunque por sí solo, el proceso de prueba no garantiza la calidad del software que se está implementado, contribuyendo a los objetivos de costo beneficio, tiempo y calidad para el final producto. Así, se recogen las características clave del proceso de prueba, recopilados "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.

(Hu et al., 2020) 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.

(J. 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.

1.1 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) que procuran 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 (ISO, s. f.), o la ISO 15504 («ISO

15504 Norma de desarrollo de software SPICE ISO/IEC 15504», s. f.) para la evaluar la madurez. Así como el enfoque del producto de software, con la ISO 9126 (*ISO 9126 - Angelfire.PDF*, s. f.) para evaluar características internas y externas (González Pinzón & González Sanabria, 2013) o como la ISO 14598 (*Queue | ISO 14598*, s. f.) con métricas y requisitos en el proceso de evaluación y la ISO 25000 (*NORMAS ISO 25000*, s. f.) 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 al., 2020) 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 en (Yu et al., 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, por lo que 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 de 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).

1.2. Teorías relacionadas al tema.

1.2.1. 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, 2009) documento que discute varias declaraciones que hasta el día de hoy se conocen como "prueba de corrección". En éste 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 mínimo del 30% según (H.

Zhang et al., 2007) al 50% del costo total del desarrollo del equipo software, como lo indica (*Software Testing Techniques - GeeksforGeeks*, s. f.),.

Algunos autores como (B. Zhang & Wang, 2011) y (Ramamoorthy et al., 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, 2012), porque el número de casos de prueba necesarios es infinito. Además, un diseño adecuado, contribuirá 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 et al., 1975) también presenta la investigación en la cual propone “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 et al., 1976). De esta forma, se han establecido posibles procedimientos para obtener datos de prueba para aplicar en cada actividad.

(Myers, 2012), propuso un modelo orientado a la detección de pruebas para “el proceso de ejecución de 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 "An empirical study on software test effort estimation" (Kafle, s. f.) y la denominada "Software Lifecycle Validation" en (Howden, 1982), que 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", que 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 & Beach, s. f.) afirma la proposición mencionada a través de la pregunta ¿El producto está correctamente construido? "Y confirme mediante" ¿Se está fabricando el producto correcto? para 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 de pruebas. El plan de prueba establecido por esta norma se basa en sus objetivos de identificar riesgos, establecer una estrategia general, definir estructuras de 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, que 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) lo define como "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 (Watts, 1989) ampliaron los principios desarrollados por Deming a la aplicación 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 el cual el autor 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, 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 otra técnica basada 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 un año fundamental, donde 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 (*Ingenieria De Software Pressman 7 Edicion Pdf*, s. f.) 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, 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 de 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, 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 la calidad, estableciendo 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, el 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 en (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 de 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.
- Económico, 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 lograrlo, se han desarrollado varios modelos de ciclo de vida, que aseguran 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 (*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.) 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", cuyo objetivo es asegurar que un producto resuelve el problema original. Luego (*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.) y (*Ingenieria De Software Pressman 7 Edicion Pdf*, s. f.) 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, se determina 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.

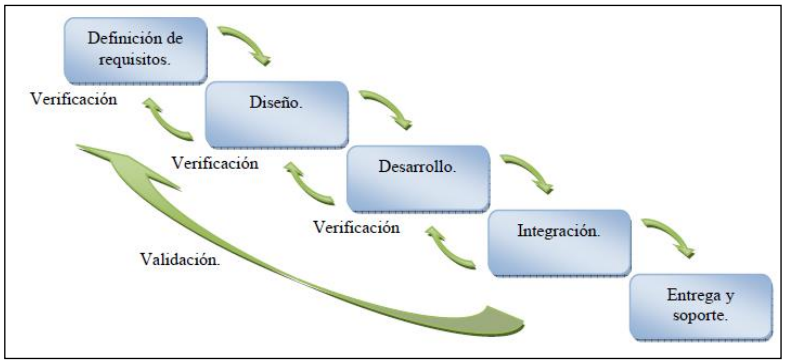


Figura 3. V&V en el ciclo de vida software

Según (*IEEE Std 610.12-1990 (R2002) - IEEE Standard Glossary of Software Engineering Terminology*, s. f.) 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 (Metzger & Pohl, 2007) 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 de Ludwind Von Bertalanffy en (Dueñas, 1989). 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 según la norma ISO 9126 en (Olivares, 2020) "...procura comprender y definir correctamente las necesidades que plantea el experto en el negocio". Para (Bayona-Oré et al., 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" aparece publicado formalmente en enero de 1990 en (*IEEE Std 610.12-1990 (R2002) - IEEE Standard Glossary of Software Engineering Terminology*, s. f.).

Del proceso de Verificación («IEEE Standard for Software Verification and Validation», 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 (*Software Engineering, 10th Edition*, s. f.) 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 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.

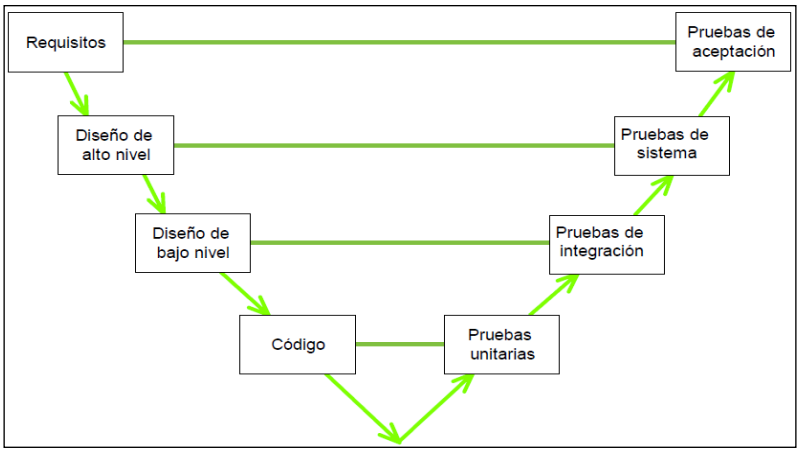


Figura 4. Proceso de desarrollo en “V”

Según la investigación (Redondo, 2002) las 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 y (b) derivación de un modelo de requisitos de software a partir de un modelo comercial. Este trabajo permite tener un sólido punto de partida para la construcción de sistemas de información, donde cada requerimiento se deriva de los objetivos comerciales.

(Nicolás Ros, 2010), en su investigación sobre la integración de modelos de análisis de dominio y requisitos del lenguaje natural, plantea la construcción de forma incremental de 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) como 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, 2008), que incluye las firmas de funciones de nivel superior del sistema y los tipos representados 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.

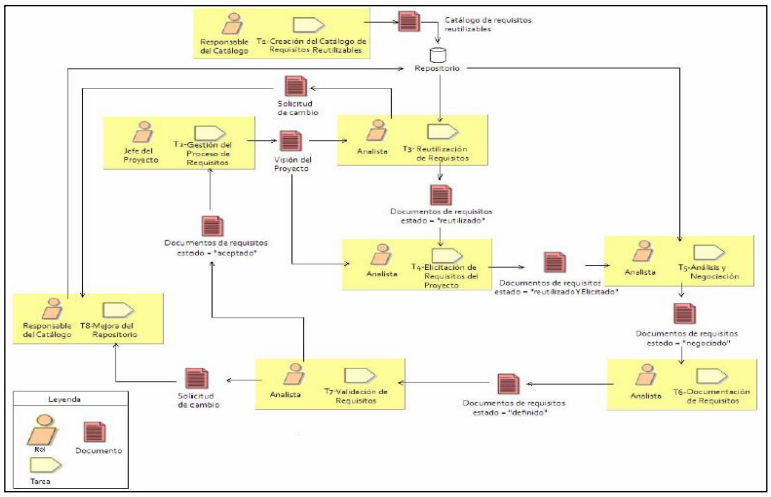


Figura 5. Proceso SIREN

1.2.2. 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 (Báez & Brunner, s. f.).

Durante los 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 consideraba un producto, al igual que los programas se desarrollaban 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 (*Ingenieria De Software Pressman 7 Edicion Pdf*, s. f.). Las características de las aplicaciones de software para ser considerado exitosas eran: a) se ejecutan, b) se ejecutan rápidamente, c) proporcionar comentarios aceptables y la 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, fue 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. (*Software Engineering, 10th Edition*, s. f.) 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 se desarrollaron para tratar el software como un producto técnico que se enmarca en fases: "planificación, análisis, diseño, implementación, pruebas y mantenimiento" (*Ingenieria De Software Pressman 7 Edicion Pdf*, s. f.).

Según (Naur et al., 1976) define 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, 1984). El término adoptado finalmente fue "Ingeniería de Software"

En la investigación (*Fundamentals of Software Engineering, 2nd Edition*, s. f.) se define la ingeniería de software 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 (*Software Requirements Engineering, 2nd Edition | Wiley*, s. f.) 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 (*Ingenieria De Software Pressman 7 Edicion Pdf*, s. f.) 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 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.".

En esta situación (Anderson & Dorfman, 1991) señalaron que: “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 (Anderson & Dorfman, 1991) 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 (*Requirements Engineering*, s. f.), el proceso de la 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 (*Ingeniería de requisitos - EcuRed*, s. f.). 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 según (Anton et al., 2001) y (Sutcliffe, 2000), así como (Benner et al., 1993), (Carroll, 1995), (Gough et al., 1995), aportan en este mismo enfoque. 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 et al., 1999).

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

En (*Ingeniería de requisitos - EcuRed*, s. f.) 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 diversas 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.

1.2.3. Marco Conceptual.

Análisis de requerimientos: Se le conoce también como "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 que 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: Es la elaboración de requisitos que incluye por un lado el contexto del sistema y, por otro, el origen de los requerimientos.

Ingeniería de Software: Según (*Ingenieria De Software Pressman 7 Edicion Pdf*, s. f.) 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.

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. Según Ackoff en (Camacho, s. f.) "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, 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 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.

1.3 Formulación del Problema.

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

1.4 Justificación e importancia del estudio.

Al verificar los requisitos de software, se han utilizado diferentes herramientas, como el desarrollo de encuestas, con el objetivo de apoyar el proceso de desarrollo de software, reduciendo las condiciones de insatisfacción de cada autor participante. Según Estrada (Estrada, 2002) 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 (Nicolás Ros, 2010) 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, luego cambiando estos en 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.

1.5 Hipótesis

Al proponer un modelo de verificación de requisitos software, que tenga en cuenta la integración de métodos de caso de prueba, se contribuye en la mejora de procedimientos del desarrollo de software comercial.

1.6 Variables.

Variable independiente:

Modelo de verificación de requisitos software.

Variable dependiente:

Procedimientos del desarrollo de software comercial.

1.7 Objetivos

1.7.1 Objetivos General

Elaborar un modelo de verificación de requisitos software, con un enfoque sistémico aplicando procesos de retroalimentación continua, como mejora de procedimientos de 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.

La **novedad científica de la investigación,** radica en revelar la lógica de la integración con un enfoque sistémico, aplicando procesos de retroalimentación continua, como mejora de procedimientos de desarrollo de software comercial.

La significación práctica, radica en el impacto social al contribuir en la mejora del seguimiento del desarrollo de software comercial, ya que el modelo de verificación de requisitos mejora los procedimientos de desarrollo de software, para reducir los niveles 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.

II. MATERIAL Y MÉTODO

2.1. Tipo y Diseño de Investigación

La presente investigación es de tipo mixta, como lo indica en libro (*Metodologia de la Investigacion - Sampieri (6ta edicion).pdf*', s. f.) que lo define como "..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 determinará el nivel de aceptación de los requisitos software por parte del experto del negocio, con las estrategias.

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.

Tabla 01. *Muestra poblacional*

Categoría	Nº
Experto en el negocio	10
Experto desarrollador de software	32
Total	**42**

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.

III. Descripción del modelo de verificación de requisitos

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:

Tabla 02. *Actividades por etapa de desarrollo de software*

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.

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, agrupándose técnicamente en funcionales y no funcionales, así como la subclasificación de estos según (*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.) 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 se clasifican en:

- Producto.
- Organización
- Externo

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

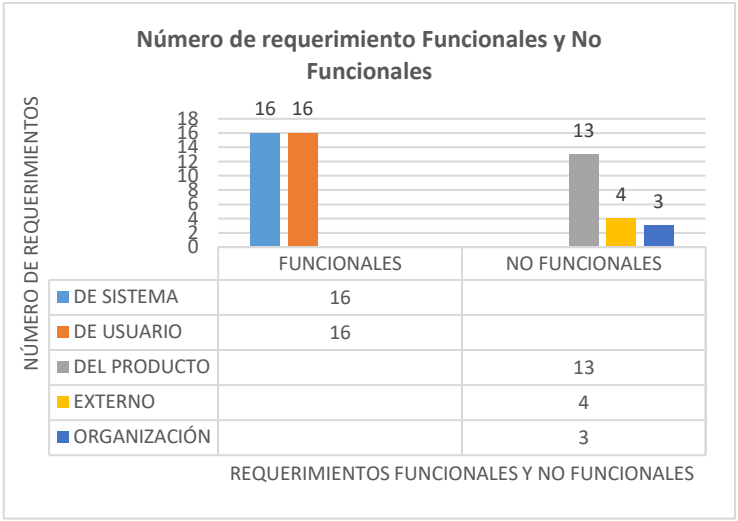


Gráfico 01. *Nivel proporcional de requerimientos funcionales y no funcionales*

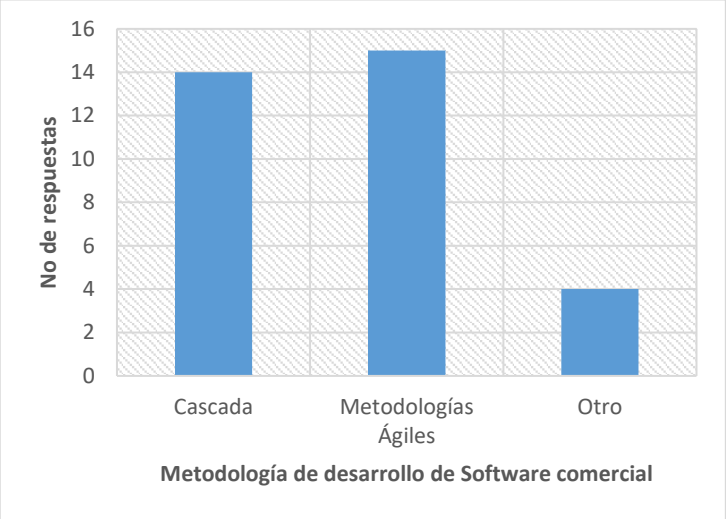
Así mismo en la pregunta, en la cual se busca conocer cuáles son los términos utilizados por los usuarios para describir sus requerimientos, planteados en su lenguaje natural cuando solicita la elaboración de una aplicación software comercial, se señalan según se indica en la Tabla 3. Siendo los procesos 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:

Tabla 03. *Número de procesos solicitados*

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

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 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 4, agrupando las frecuencias en alta, media y baja, 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.

Tabla 04. *Nivel de frecuencia de requerimientos interpretados equívocamente por el experto desarrollador de software*

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%

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 un modelo de verificación de requisitos software, con un enfoque sistémico de procesos de retroalimentación continua, para el seguimiento de desarrollo de software comercial.

Tabla 05. *Causales de una incorrecta interpretación de requisitos por parte del experto desarrollador de software.*

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%

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.

3.1 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.

3.1.1. 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 (*Ingeniería del software - Introducción a la ingeniería del software PID_ Jordi Pradel Miquel Jose*, s. f.) 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 (*(Global Edition) Ian Sommerville - Software Engineering, 10th Edition-Pearson (2016).pdf*, s. f.) 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, 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 es 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 Std 610.12-1990 (R2002) - IEEE Standard Glossary of Software Engineering Terminology*, s. f.) indica que "… usualmente son consideradas para validación; pero también para verificación", y sobre las pruebas de software en su estudio (Myers, 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.

3.1.2. 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:

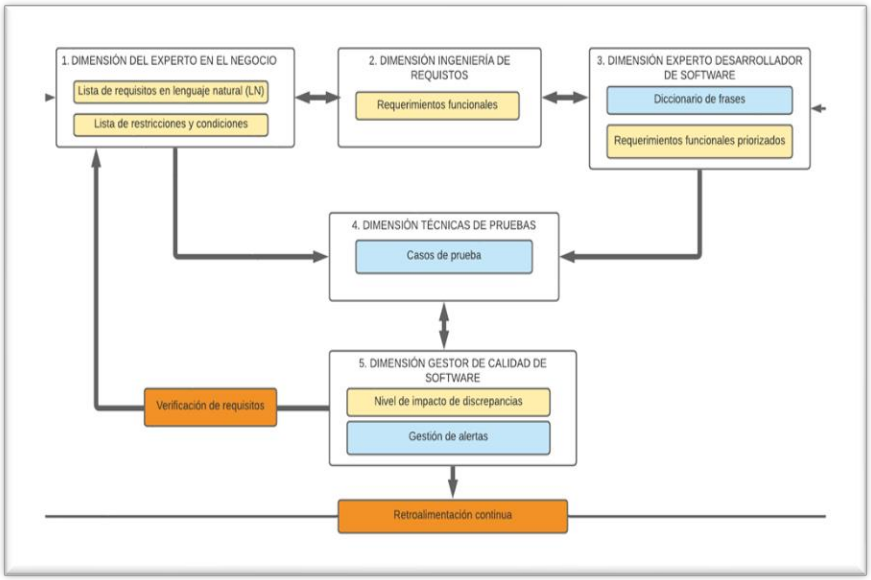


Figura 6. Modelo de verificación de requisitos

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, generando 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 que, 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 de mejora continua, 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 dando a conocer 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 (Olivares, 2020) 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 (*IEE_STD_1233-_Requirements_Spec.pdf*, s. f.) 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 Standard for Software Verification and Validation», 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, inspección, evaluación y prueba de productos y procesos de software".

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, los cuales son 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 evaluar 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

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.

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

About ACM Resources. (s. f.). Recuperado 12 de junio de 2022, de https://learning.acm.org/resources

Anderson, C., & Dorfman, M. (1991). *Aerospace Software Engineering: A Collection Of Concepts*. https://doi.org/10.2514/4.866098

Anton, A. I., Earp, J. B., Potts, C., & Alspaugh, T. A. (2001). The role of policy and stakeholder privacy values in requirements engineering. *Proceedings Fifth IEEE International Symposium on Requirements Engineering*, 138-145. https://doi.org/10.1109/ISRE.2001.948553

Ayabaca, L., & Moscoso Bernal, S. (2018). Verificación, validación y pruebas de software. *Killkana Técnica*, *1*, 29. https://doi.org/10.26871/killkana_tecnica.v1i3.112

Báez, M. G., & Brunner, S. I. B. (s. f.). *Metodología DoRCU para la Ingeniería de Requerimientos*. 13.

Bayona-Oré, S., Chamilco, J., & Perez, D. (2019). Mejora de Procesos Software: Gestión de Requisitos, Verificación y Validación: Software Process Improvement: Requirements Management, Verification and Validation. *CISTI (Iberian Conference on Information Systems & Technologies / Conferência Ibérica de Sistemas e Tecnologias de Informação) Proceedings*, 1-5.

Benner, K. M., Feather, M. S., Johnson, W. L., & Zorman, L. A. (1993). Utilizing Scenarios in the Software Development Process. En N. Prakash, C. Rolland, & B. Pernici (Eds.), *Information System Development Process* (pp. 117-134). North-Holland. https://doi.org/10.1016/B978-0-444-81594-1.50013-1

Bird, D. L., & Munoz, C. U. (1983). Automatic generation of random self-checking test cases. *IBM Systems Journal*, *22*(3), 229-245. https://doi.org/10.1147/sj.223.0229

Boehm, B. W., & Beach, R. (s. f.). *SOFTWARE ENGINEERING - AS IT IS*. 11.

Boyer, R. S., Elspas, B., & Levitt, K. N. (1975). SELECT—a formal system for testing and debugging programs by symbolic execution. *ACM SIGPLAN Notices*, *10*(6), 234-245. https://doi.org/10.1145/390016.808445

Camacho, A. O. (s. f.). *El arte de resolver problemas ackoff (5)*. Recuperado 21 de julio de 2022, de https://www.academia.edu/39676018/El_arte_de_resolver_problemas_ackoff_5_

Canfora, G. (2004). Software Evolution in the Era of Software Services. *Proceedings of the Principles of Software Evolution, 7th International Workshop*, 9-18.

Carrizo, D., & Rojas, J. (2018). Metodologías, técnicas y herramientas en ingeniería de requisitos: Un mapeo sistemático. *Ingeniare. Revista chilena de ingeniería*, *26*(3), 473-485. https://doi.org/10.4067/S0718-33052018000300473

Carroll, J. M. (1995). *Scenario-based design: Envisioning work and technology in system development*. https://typeset.io/papers/scenario-based-design-envisioning-work-and-technology-in-2366tpuzae

Clarke, L. A. (1976). A System to Generate Test Data and Symbolically Execute Programs. *IEEE Transactions on Software Engineering*, *SE-2*(3), 215-222. https://doi.org/10.1109/TSE.1976.233817

Dueñas, C. (1989). Teoria general de los sistemas—Ludwigvon Bertalanffy. *Teoria General de Los*

Sistemas. https://www.academia.edu/17445870/Teoria_general_de_los_sistemas_Ludwigvon_Bertalanffy

Estrada, H. (2002). *Generación de Especificaciones de Requisitos de Software a partir de Modelos de Negocios: Un enfoque basado en metas*. https://www.academia.edu/67002484/Generaci%C3%B3n_de_Especificaciones_de_Requisitos_de_Software_a_partir_de_Modelos_de_Negocios_un_enfoque_basado_en_metas

Fundamentals of Software Engineering, 2nd Edition. (s. f.). Recuperado 16 de julio de 2022, de https://www.pearson.com/content/one-dot-com/one-dot-com/us/en/higher-education/program.html

George, C., & Haxthausen, A. E. (2008). The Logic of the RAISE Specification Language. En D. Bjørner & M. C. Henson (Eds.), *Logics of Specification Languages* (pp. 349-399). Springer. https://doi.org/10.1007/978-3-540-74107-7_7

(Global Edition) Ian Sommerville—Software Engineering, 10th Edition-Pearson (2016).pdf. (s. f.). Recuperado 13 de junio de 2022, de https://docs.google.com/viewer?a=v&pid=sites&srcid=dWhkLmVkdS5pcXxob2dlci1tYWhtdWR8Z3g6NTYwZWVjNWM3NzdjMDJlZA

González Pinzón, M. F., & González Sanabria, J. S. (2013). Aplicación del estándar ISO/IEC 9126-3 en el modelo de datos conceptual entidad-relación. *Revista Facultad de Ingeniería*, *22*(35), 113-125.

Gough, P., Fodemski, F., Higgins, S. A., & Ray, S. J. (1995). *Scenarios-an industrial case study and hypermedia enhancements* (p. 17). https://doi.org/10.1109/ISRE.1995.512541

Hadad, G., Jorge Horacio, D., Kaplan, G., & Leite, J. (1999). *Enfoque Middle-Out en la Construcción e Integración de Escenarios*. (p. 94).

Hoare, C. A. R. (1984). Programming: Sorcery or Science? *IEEE Software*, *1*(2), 5-16. https://doi.org/10.1109/MS.1984.234042

Howden, W. E. (1982). Life-Cycle Software Validation. *Computer*, *15*(02), 71-78. https://doi.org/10.1109/MC.1982.1653943

Hu, X., Zhuang, Y., & Zhang, F. (2020). A security modeling and verification method of embedded software based on Z and MARTE. *Computers & Security*, *88*, 101615. https://doi.org/10.1016/j.cose.2019.101615

IEEE Standard for Software Test Documentation. (1983). *IEEE Std 829-1983*, 1-48. https://doi.org/10.1109/IEEESTD.1983.81615

IEEE Standard for Software Unit Testing. (1986). *ANSI/IEEE Std 1008-1987*, 1-28. https://doi.org/10.1109/IEEESTD.1986.81001

IEEE Standard for Software Verification and Validation. (2005). *IEEE Std 1012-2004 (Revision of IEEE Std 1012-1998)*, 1-110. https://doi.org/10.1109/IEEESTD.2005.96278

IEEE Std 610.12-1990 (R2002)—IEEE Standard Glossary of Software Engineering Terminology. (s. f.). Recuperado 28 de noviembre de 2021, de https://webstore.ansi.org/standards/ieee/ieeestd610121990r2002

IEE_STD_1233-_Requirements_Spec.pdf. (s. f.). Recuperado 17 de diciembre de 2021, de https://docs.google.com/viewer?a=v&pid=sites&srcid=bWl4Lnd2dS5lZHV8Y3NlZTQ4MHxneDozODFlNTk0NzY1NGQ3NDk2

Ingeniería de requisitos—EcuRed. (s. f.). Recuperado 20 de septiembre de 2021, de

https://www.ecured.cu/Ingenier%C3%ADa_de_requisitos

Ingenieria De Software Pressman 7 Edicion Pdf. (s. f.). Recuperado 8 de septiembre de 2021, de https://readwritesoar.com/ingenieria-de-software-pressman-7-edicion-pdf.html

Ingeniería del software. Un enfoque práctico, 7ma Edición – Roger S. Pressman. (2020, marzo 13). *BLOG*. https://bloginfotecs.com/ingenieria-del-software-un-enfoque-practico-7ma-edicion-roger-s-pressman/

Ingeniería del software—Introducción a la ingeniería del software PID_ Jordi Pradel Miquel Jose. (s. f.). StuDocu. Recuperado 26 de julio de 2022, de https://www.studocu.com/ca-es/document/universitat-oberta-de-catalunya/ingenieria-del-software/ingenieria-del-software/24087131

ISO. (s. f.). *ISO/IEC/IEEE 12207:2017*. ISO. Recuperado 26 de junio de 2022, de https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/37/63712.html

ISO 9126—Angelfire.PDF. (s. f.). Recuperado 28 de noviembre de 2021, de https://motor-busqueda-libros.com/iso-9126-angelfire-pdf-dl222211

ISO 15504 Norma de desarrollo de software SPICE ISO/IEC 15504. (s. f.). *Normas ISO*. Recuperado 23 de noviembre de 2021, de https://www.normas-iso.com/iso-iec-15504-spice/

Kafle, L. P. (s. f.). *AN EMPIRICAL STUDY ON SOFTWARE TEST EFFORT ESTIMATION*. *2*(2), 11.

Korel, B. (1990). Automated software test data generation. *IEEE Transactions on Software Engineering*, *16*(8), 870-879. https://doi.org/10.1109/32.57624

LAS MIPYME EN CIFRAS 2016. (s. f.). Recuperado 15 de junio de 2022, de https://ogeiee.produce.gob.pe/index.php/en/shortcode/oee-documentos-publicaciones/publicaciones-anuales/item/758-las-mipyme-en-cifras-2016

LAS MIPYME EN CIFRAS 2020. (s. f.). Recuperado 17 de junio de 2022, de https://ogeiee.produce.gob.pe/index.php/en/shortcode/oee-documentos-publicaciones/publicaciones-anuales/item/1008-las-mipyme-en-cifras-2020

Metodologia de la Investigacion—Sampieri (6ta edicion).pdf. (s. f.). Google Docs. Recuperado 17 de julio de 2022, de https://drive.google.com/file/d/0B7fKI4RAT39QeHNzTGh0N19SME0/view?usp=sharing&usp=embed_facebook

Metzger, A., & Pohl, K. (2007). Variability Management in Software Product Line Engineering. *29th International Conference on Software Engineering (ICSE'07 Companion)*, 186-187. https://doi.org/10.1109/ICSECOMPANION.2007.83

Mohammed, A., Alsarraj, R., & Albayati, A. (2020). VERIFICATION AND VALIDATION OF A SOFTWARE: A REVIEW OF THE LITERATURE. *Iraqi Journal for Computers and Informatics*, *46*, 40-47. https://doi.org/10.25195/ijci.v46i1.249

Myers, G. J. (2012). *The Art of Software Testing*. John Wiley & Sons.

Naur, P., Randell, B., Buxton, J. N., NATO, NATO, OTAN, & Working Conference on Software Engineering Techniques, C. on S. E. (Eds.). (1976). *Software*

engineering: Concepts and techniques : proceedings of the NATO conference. Petrocelli.

Nicolás Ros, J. (2010). *Una Propuesta de Gestión Integrada de Modelos y Requisitos en Líneas de Productos Software*.

NORMAS ISO 25000. (s. f.). Recuperado 28 de noviembre de 2021, de https://iso25000.com/index.php/normas-iso-25000

Olivares, S. G. D. (2020). Iso 9126. *Calidad de Software : Iso 9126*. https://www.academia.edu/43099704/Iso_9126

Özakıncı, R., & Tarhan, A. (2018). Early software defect prediction: A systematic map and review. *Journal of Systems and Software*, *144*, 216-239. https://doi.org/10.1016/j.jss.2018.06.025

Queue | ISO 14598. (s. f.). Vsip.Info. Recuperado 28 de noviembre de 2021, de https://vsip.info/qdownload/iso-14598-pdf-free.html

Ramamoorthy, C. V., Ho, S.-B. F., & Chen, W. T. (1976). On the Automated Generation of Program Test Data. *IEEE Transactions on Software Engineering*, *SE-2*(4), 293-300. https://doi.org/10.1109/TSE.1976.233835

Redondo, R. D. (2002). *Reutilización de requisitos funcionales de sistemas distribuidos utilizando técnicas de descripción formal* [Http://purl.org/dc/dcmitype/Text, Universidade de Vigo]. https://dialnet.unirioja.es/servlet/tesis?codigo=10527

Requirements Engineering: Processes and Techniques | Wiley. (s. f.). Wiley.Com. Recuperado 17 de julio de 2022, de https://www.wiley.com/en-

us/Requirements+Engineering%3A+Processes+and+Techniques-p-9780471972082

Sabadell, X. E. (s. f.). *Calidad del software: Gestión de la calidad y métricas*. 90.

Sánchez Peño, J. M. (2015, junio 16). *Pruebas de Software. Fundamentos y Técnicas* [Info:eu-repo/semantics/bachelorThesis]. E.T.S.I y Sistemas de Telecomunicación (UPM). https://oa.upm.es/40012/

Sawant, A., Bari, P., & Chawan, P. (2012). Software Testing Techniques and Strategies. *International Journal of Engineering Research and Applications(IJERA)*, *2*, 980-986.

Software Engineering, 10th Edition. (s. f.). Recuperado 12 de junio de 2022, de https://www.pearson.com/content/one-dot-com/one-dot-com/us/en/higher-education/program.html

Software Requirements Engineering, 2nd Edition | Wiley. (s. f.). Wiley.Com. Recuperado 17 de julio de 2022, de https://www.wiley.com/en-us/Software+Requirements+Engineering%2C+2nd+Edition-p-9780818677380

Software Testing Techniques—GeeksforGeeks. (s. f.). Recuperado 20 de noviembre de 2021, de https://www.geeksforgeeks.org/software-testing-techniques/

Souza, A. (s. f.). *A View of 20th and 21st Century Software Engineering*. Recuperado 15 de junio de 2022, de https://www.academia.edu/es/34858377/A_View_of_20th_and_21st_Century_Software_Engineering

Sutcliffe, A. (2000). Domain analysis for software reuse. *Journal of Systems and Software*, *50*(3), 175-199. https://doi.org/10.1016/S0164-1212(99)00096-5

The complete guide to software testing (world). (s. f.). Guide Books. Recuperado 11 de julio de 2022, de https://dl.acm.org/doi/abs/10.5555/42384

Turing, A. M. (2009). Computing Machinery and Intelligence. En R. Epstein, G. Roberts, & G. Beber (Eds.), *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer* (pp. 23-65). Springer Netherlands. https://doi.org/10.1007/978-1-4020-6710-5_3

W, L. (1989). Managing the Software Process. *Journal of Information Technology*, *4*(3), 172-172. https://doi.org/10.1057/jit.1989.28

Wang, Y., & Wang, Y. (2008). Software Process in Software Project Management. *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02*, 594-596. https://doi.org/10.1109/CSSE.2008.297

Yamada, S., Omori, T., & Ohnishi, A. (2019). Verification method of reliability requirements. *Procedia Computer Science*, *159*, 860-869. https://doi.org/10.1016/j.procs.2019.09.245

Yu, C., Li, Q., Liu, K., Chen, Y., & Wei, H. (2021). Industrial Design and Development Software System Architecture Based on Model-Based Systems Engineering and Cloud Computing. *Annual Reviews in Control*, *51*, 401-423. https://doi.org/10.1016/j.arcontrol.2021.04.011

Zhang, B., & Wang, C. (2011). Automatic generation of test data for path testing by adaptive genetic simulated annealing algorithm. *2011 IEEE International Conference on Computer Science and Automation Engineering*, *2*, 38-42. https://doi.org/10.1109/CSAE.2011.5952418

Zhang, H., Kitchenham, B., & Jeffery, R. (2007). A Framework for Adopting Software Process

Simulation in CMMI Organizations. En Q. Wang, D. Pfahl, & D. M. Raffo (Eds.), *Software Process Dynamics and Agility* (pp. 320-331). Springer. https://doi.org/10.1007/978-3-540-72426-1_27

Zhang, J., & Li, J. (2020). Testing and verification of neural-network-based safety-critical control software: A systematic literature review. *Information and Software Technology*, *123*, 106296. https://doi.org/10.1016/j.infsof.2020.106296

ISBN: 978-9942-33-621-7

compAs
Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com