Universidad Nacional Pedro Ruiz Gallo

FACULTAD DE INGENIERÍA CIVIL, DE SISTEMAS Y ARQUITECTURA

FACULTAD PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

"Método para la Gestión del Desarrollo del Software, basada en la NTP ISO/IEC 12207:2006 en la Municipalidad Provincial de Chiclayo"

Para obtener el Título Profesional de:

Ingeniera de Sistemas

Lozano Alarcón Claudia Celeste

Montenegro Risco Diana Carolina

Autores

M.sc. Ing. Ernesto Karlo Celi Arévalo

Asesor

Noviembre 2020

Lambayeque, Perú

Universidad Nacional Pedro Ruiz Gallo

FACULTAD DE INGENIERÍA CIVIL, DE SISTEMAS Y ARQUITECTURA

FACULTAD PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

"Método para la Gestión del Desarrollo del Software, basada en la NTP ISO/IEC 12207:2006 en la Municipalidad Provincial de Chiclayo"

Para obtener el Título Profesional de:

Ingeniera de Sistemas

Aprobado por los miembros del jurado:

ING. FRANK RICHARD RODRIGUEZ CHIRINOS

Presidente del Jurado

ING. OSCAR EFRAIN CAPUÑAY UCEDA

Vocal del Jurado

ING. CESAR AUGUSTO GUZMAN VALLE

Secretario del Jurado

M.SC. ING. ERNESTO KARLO CELI ARÉVALO

Asesor de Tesis

MONTENEGRO RISCO DIANA CAROLINA

Autor

LOZANO ALARCÓN CLAUDIA CELESTE

Autor

DEDICATORIA

A Dios, por darme su bendición día a día y por ser el camino que me guía en cada paso que doy y a mi familia que son la fuente de motivación y de superación en mí vida.

AGRADECIMIENTO

Agradezco a mi Asesor por el apoyo, paciencia, tiempo y por los conocimientos brindados para el desarrollo de Tesis.

A mis grandes amigos y compañeros de aulas por todo el apoyo y buenos momentos compartidos y a mis profesores que contribuyeron en mi formación profesional.

RESUMEN

La presente tesis aborda la problemática que presenta la Municipalidad Provincial de Chiclayo en la sección de Desarrollo de Software que es la encargada de desarrollar e implementar los sistemas de información que requiere la Institución.

Actualmente esta función no es llevada a cabo óptimamente ya que no cuenta con sus procesos debidamente documentados, cabe indicar que al carecer de una procedimentación se estaría incumpliendo con las directrices y políticas establecidas por la oficina de sistemas de la presidencia del consejo de ministros (PCM), en el marco de la NTP ISO/IEC 12207, siendo de uso obligatorio en las entidades públicas. La pretensión de esta tesis fue desarrollar un método para la gestión del desarrollo del software basada en la NTP ISO 12207:2006, que permita mejorar el proceso de desarrollo de software con el objetivo de disminuir errores, asignar responsabilidades, reducir la incidencia de cambios emitidos por los usuarios en las etapas finales y así controlar los tiempos de entrega y desarrollo del software, evitando usuarios insatisfechos y baja calidad en los sistemas.

Palabras clave: ISO/IEC 12207, Metodología de desarrollo de software, ciclo de vida del software.

ABSTRACT

This thesis addresses the problem presented by the Provincial Municipality of Chiclayo in the Software Development section that is responsible for developing and implementing the information systems required by the Institution.

Currently, this function is not carried out optimally since it does not have its processes duly documented, it should be noted that, lacking a procedure, it would be in breach of the guidelines and policies established by the systems office of the presidency of the council of ministers (PCM), within the framework of the NTP ISO / IEC 12207, being mandatory in public entities. The aim of this thesis was to develop a method for the management of software development based on the NTP ISO 12207: 2006, which allows to improve the software development process with the objective of reducing errors, assigning responsibilities, reducing the incidence of changes issued by the users in the final stages and thus control the delivery and development times of the software, avoiding unsatisfied users and quality in the systems.

Key Word: ISO / IEC 12207, Software development methodology, software life cycle.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE DE ANEXOS

INTRODUCCION

Actualmente el éxito de los proyectos de desarrollo de software está relacionado con una serie de factores uno de ellos es seguir una línea metodológica, la cual sirva de guía para llevar el control y documentar el avance del desarrollo de software, hoy en día podemos encontrar en una serie de marcos de referencia dentro de los que destacan la ISO 12207, que es la encargada de la ingeniería y del apoyo en el proceso de desarrollo de software siendo esta de uso obligatorio, en las entidades públicas. Según Resolución Ministerial N.º 004-2017-PCM.

Es por ello que la presente investigación ha sido dividida en seis capítulos descritos de la siguiente manera:

En el primer capítulo se realizó una descripción de la situación problemática de cómo se encuentra actualmente el proceso de desarrollo de software en la MPCH, así como la justificación e importancia de la presente investigación.

En el segundo capítulo se describió todo lo referente a la fundamentación teórica que contiene información necesaria sobre el proceso completo de desarrollo de software, se dan a conocer algunos modelos aplicables al ciclo de vida del software, la importancia y modo de evaluación de la Norma ISO/IEC 25010 y la ISO 12207:2017 la cual permite abordar adecuadamente el problema.

En el tercer capítulo especificó el marco metodológico, que tipo de investigación se realizó y los resultados que se esperó obtener de este trabajo de investigación.

En el cuarto capítulo se describió la presentación de la propuesta para la solución de la situación problemática.

En el quinto capítulo se detalló los resultados obtenidos de la evaluación de los procesos del ciclo de vida del software propuestos en el desarrollo de la propuesta.

En el sexto capítulo se presentan las conclusiones y recomendaciones obtenidas en este trabajo de investigación.

CAPÍTULO I: Plan de la investigación

1.1. Situación problemática

La presente tesis aborda la problemática que presenta la Municipalidad Provincial de Chiclayo (de ahora en adelante MPCH), en la sección de Desarrollo de Sistemas la cual se encuentra dentro de la Gerencia de Sistemas e Informática.

La sección de Desarrollo de Sistemas es la encargada de desarrollar e implementar los sistemas de información que requiere la Institución , además de actualizar y mantener el buen funcionamiento de aquellos sistemas que están en producción, para ello es necesario cumplir con el proceso de ciclo de vida del software, los cuales tienen que estar debidamente documentados según la Resolución Ministerial N.º 041-2017-PCM; sin embargo esto no se lleva a cabo de una manera óptima debido a que dicha área no cuenta con una procedimentación para llevar a cabo la documentación del software; ni con una persona que se encargue de ello, solo cuenta con dos trabajadores quienes únicamente se encargan del desarrollo del software.

Actualmente, el proceso de desarrollo de nuevos sistemas informáticos de la Gerencia de Sistemas e Informática según el manual de procesos (MAPRO), con fecha del 24 de agosto del 2010, que se encuentra en el portal web de la institución está dado de la siguiente manera:

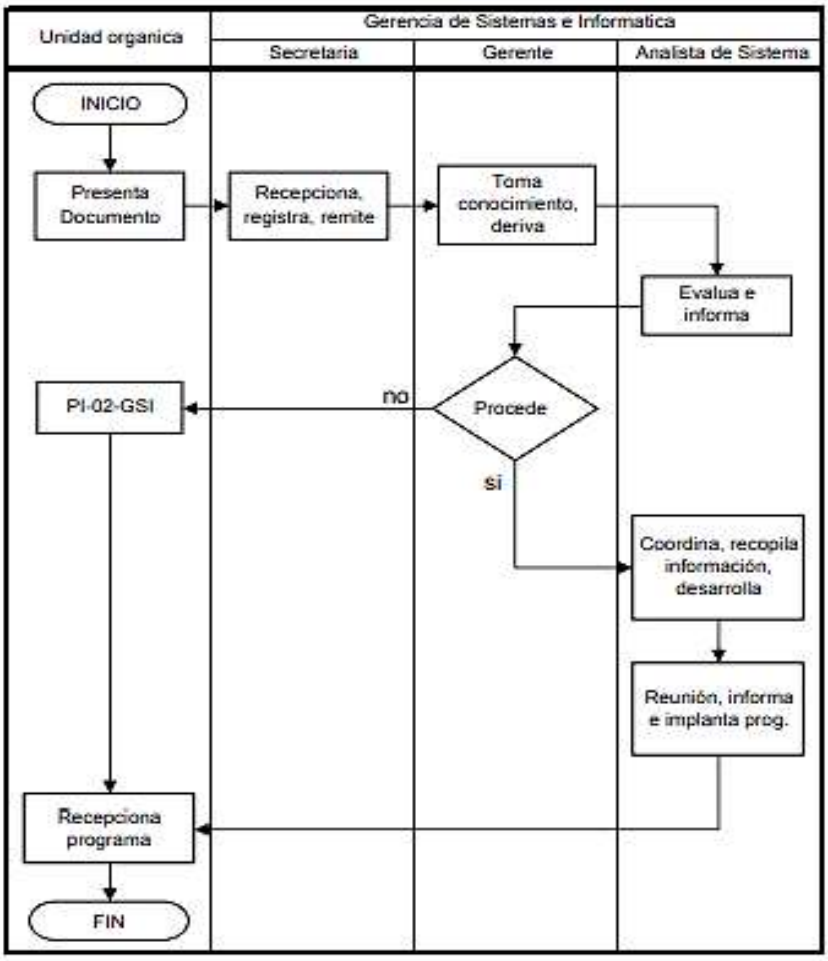


Ilustración 1: Diagrama de flujo del Proceso de desarrollo de nuevos sistemas informáticos
Fuente: (MPCH, 2010)

REQUISITOS - ANTECEDENTES	N°	UNIDAD ORGANICA	ACCIÓN	DURACIÓN
Solicitud de requerimiento dirigida al Gerente de Sistemas e Informática	1	Unidad organica	1° Presenta documento	
	2	Gerencia de Sistema e Informatica	1° Secretaria: Recepciona solicitud, registra y remite documento al Gerente de Sistemas e informática	5 min.
			2° Gerente: Toma conocimiento y provee a Analista de sistemas	15 min.
			3° Analista de sistemas: Evalua requerimiento e informa a Gerencia si desarrolla o contrata a terceros	2 d
			4° Gerente: Coordina, aprueba desarrollo del sistema	1 d
			5° Analista de sistemas: Coordina trabajo con personal a su cargo, recopilación de información, reunion periodica con Gerente y usuarios, Informa trabajo terminado, procede a implantar el sistema, entrega sistema desarrollado a Gerencia de Sistemas, Informe final	116 días
	3	Unidad organica	1° Da conformidad	
			TOTAL	4 mes

Ilustración 2: Actividades del Proceso de desarrollo de nuevos sistemas informáticos
Fuente: (MPCH, 2010)

Lo descrito anteriormente, genera posteriores inconvenientes entre el desarrollador y el área usuaria, debido a que este emite cambios constantes siendo algunos de ellos básicos y de vital importancia para el desarrollo del software y que muchas veces son comunicados en una etapa muy avanzada, trayendo como consecuencia que el software no se realice en el tiempo estimado generándole insatisfacción al usuario; es por ello, que actualmente se ayudan con historias de usuario, que les permiten interactuar con el área usuaria las veces que sean necesarias después de cada avance, para que los requerimientos queden plasmados correctamente.

En base a la problemática descrita, luego de haber tenido una entrevista directamente con el jefe del área y también con los trabajadores responsables de la sección, se llegó a la conclusión es necesario y prioritario mejorar el proceso de Desarrollo de Software dado que, a través de este proceso se brinda el servicio de atención a las solicitudes de cambio en los sistemas que están en producción; así como el desarrollo de nuevos requerimientos, a más de las 40 áreas usuarias que orgánicamente cuenta la municipalidad, convirtiéndose en proceso crítico de la Sección de Desarrollo de Software. Si se maneja de forma adecuada puede hacer más eficiente la prestación de los servicios que ofrece la organización.

El proceso de desarrollo de software en la Sección de Desarrollo de Software de la municipalidad se caracteriza por las siguientes carencias o debilidades:

a) Una procedimentación: esto dificulta tener el control de cambios de los sistemas, manejar problemas de testeo y liberación de versiones.
b) Control de Cambios: nos permite gestionar apropiadamente los cambios solicitados por el usuario y sobre todo a tiempo para así no retrasar el desarrollo del sistema ni la entrega final.
c) Problemas de testeo: consisten en probar detalladamente cada etapa del proyecto de software pretendiendo que este funcione tal y como lo requiere el usuario y no contenga errores, pero generalmente los desarrolladores no le dan la importancia que merece al testeo del sistema.
d) Una documentación: que es la permite evaluar el progreso del proyecto comprobando si se adecuan o no a los requisitos funcionales. Esto puede servir, además, para la toma de decisiones a lo largo del desarrollo del proyecto. Por eso al no contar con la documentación necesaria en cada fase del ciclo de vida del software no se podría medir el progreso por lo

mismo no se tendría un control de los tiempos y la calidad de los sistemas una vez finalizado el desarrollo del mismo.

1.2. Formulación del Problema

¿De qué manera un método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2006, mejora el proceso de desarrollo de software en la Municipalidad Provincial de Chiclayo?

1.2.1. Problemas específicos

a) ¿De qué forma el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, se adecúa funcionalmente a las tareas y objetivos del proceso de desarrollo de software en la Municipalidad?
b) ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, permite niveles de usabilidad aceptables, tanto por los usuarios de TI como del personal de desarrollo del área de TI, en la Municipalidad?

a. ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 permite niveles aceptables de rendimiento para proporcionar adecuados tiempos de respuesta y procesamiento en las aplicaciones que están en producción en la Municipalidad?
c) ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 permite niveles de fiabilidad a los usuarios en el funcionamiento diario y operatividad cuando ocurran fallos en las aplicaciones que están en producción en la Municipalidad?

d) ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 permite niveles aceptables de mantenibilidad de las aplicaciones que están en producción en la Municipalidad?

1.3. Limitaciones de la Investigación

- Dificultad para acceder a determinada información a acerca de los sistemas por tema de confidencialidad.

- Dificultad para obtener información bibliográfica debido que, para acceder a algunos artículos científicos relacionados a la ISO 12207:2017, ya que es una versión más reciente y requerían de un costo que estaban fuera del alcance de nuestras posibilidades.

1.4. Objetivos de la Investigación

1.4.1. Objetivo General

Desarrollar un método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2006 que permita mejorar el proceso de desarrollo de software en la Municipalidad Provincial de Chiclayo.

1.4.2. Objetivos Específicos

b. Modelar el proceso de desarrollo de software bajo la perspectiva de la ISO 12207:2017 para determinar fases actividades y tareas.
c. Definir o elaborar las fichas técnicas de cada proceso para identificar sus entradas y salidas.
d. Elaborar los formatos que serán elaborados en cada a uno de los procesos.
e. Validar el método propuesto a través de un juicio de expertos.
f. Demostrar que método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 se adecúa funcionalmente a las tareas y objetivos del proceso de desarrollo de software en la Municipalidad Provincial de Chiclayo.
g. Demostrar que el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, permite niveles de usabilidad aceptables, tanto por los usuarios de TI como del personal de desarrollo del área de TI, en la Municipalidad.
h. Demostrar que el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 permite niveles aceptables de mantenibilidad de las aplicaciones que están en producción en la Municipalidad.
i. Demostrar que el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, permite niveles de fiabilidad a los usuarios en el funcionamiento diario y operatividad cuando ocurran fallos en las aplicaciones que están en producción en la Municipalidad.
j. Demostrar que el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, permite niveles aceptables

de rendimiento para proporcionar adecuados tiempos de respuesta y procesamiento en las aplicaciones que están en producción en la Municipalidad.

CAPÍTULO II: MARCO TEÓRICO

2.1. Estado del arte

La creación del software es un proceso intrínsecamente creativo y la ingeniería de software trata de sistematizar este proceso con el fin de reducir el riesgo del fracaso en la consecución del objetivo por medio de diversas técnicas que se han demostrado adecuadas en base a la experiencia previa.

Un objetivo para la Ingeniería de Software desde hace varias décadas ha sido el encontrar procesos y metodologías sistemáticas, predecibles y repetibles, a fin de mejorar la productividad en el desarrollo y la calidad del producto software.

Para el ciclo de desarrollo y vida del software existen diversos modelos, por ejemplo, el modelo de Cascada pura o secuencial, el modelo en Espiral, el modelo Incremental, modelo evolutivo.

Existe un gran número de empresas que desean mejorar sus procesos de desarrollo, pero no tienen la capacidad necesaria para utilizar modelos completamente nuevos por lo que requieren hacer uso de metodologías tradicionales modificadas para su beneficio.

Según la Ing. Rosa Andrea Rea Lozada y Gabriela Espinoza García para cada proyecto se debería elegir un modelo apropiado de acuerdo a las necesidades de las instituciones, en ocasiones cabe la posibilidad de combinar varios modelos, que permita asegurar la calidad del producto de software. Y así evitar una inadecuada gestión de requerimientos, fallos en la elicitación y análisis, especificación, verificación, control, seguimiento y documentación de los requerimientos; todo ello implica que exista un grado de inexactitud para la estimación de tiempo, esfuerzo y complejidad, además incidían en la existencia de documentos de requerimientos incorrectos, incompletos, no estandarizados, o no existentes.

Es por ello que ambas autoras recomiendan utilizar como base modelos como el de cascada, espiral, incremental y evolutivo, siendo el más conocido y usado el modelo de cascada por su enfoque metodológico que ordena las etapas del proceso para el desarrollo del software.

Desde el punto de vista de Edder Martin Olivares Palacios implementar un modelo de procesos basados en la ISO/IEC 12207:2008, permite definir a un nivel más específico las necesidades del área usuaria, también permite utilizar este modelo como base para la mejora de los procesos en las instituciones, así como la optimización de los procesos en términos de aumento de la producción, reducción de costos, incremento de la calidad y de la satisfacción del cliente.

Así mismo Sosa Carhuamaca Nataly Jenny, Evelyn Joanna Chinarro Morales y Nathaly María Llalleri Cardenas consideran que el proceso de desarrollo del ciclo de vida del software debe darse en base a las pautas definidas por la Norma Técnica Peruana ISO/IEC 12207:2016, siendo ésta una versión más actual, la cual permite mejorar la gestión de desarrollo de software, dado que es necesario que el proceso tenga un nivel de cumplimiento de buenas prácticas, metodologías y técnicas aplicadas, debido a que la escasez de una metodología genera que los procesos y actividades no se encuentren debidamente formalizados ni documentados y carecerían de un control, además de invertir más tiempo y recursos para subsanar los errores no encontrados en los sistemas, al mismo tiempo ocasiona que los usuarios reporten incidencias con frecuencia las cuales retrasan el óptimo desarrollo de sus funciones.

Actualmente, el desarrollo de las TIC's influye en gran escala con el desarrollo social y económico de una nación, por tal motivo las instituciones desarrolladoras de software se ven en la necesidad de establecer una metodología; pero no es suficiente con definir un modelo existente si este no está ejecutado de una forma adecuada, es por eso que existe la necesidad de utilizar estándares que permiten la evaluación de los procesos de desarrollo del software.

Para ello la Ing. Baldeón Villanes Edú James proponen método para la evaluación de calidad de software basado en ISO/IEC 25000, el cual fue aplicado a una muestra representativa de proyectos, llegando a demostrar que el software durante su ciclo de vida mejoró la calidad del producto final, facilitando la conformidad por parte del usuario y disminución de los errores después de su puesta en producción. Por lo tanto, se mostró que la evaluación del proceso de desarrollo de software permite determinar el nivel de capacidad de los procesos a través del cumplimiento de las buenas prácticas, y que es de vital importancia para mejorar los procesos actuales.

Según Karla Betzaida Chambilla Carazas, la utilización de la ISO/IEC 25000 permite aumentar la calidad del producto final, estableciendo puntos de control en cada uno de los entregables, logrando terminar en tiempo y forma el sistema. Además, se mejoran los resultados de calidad en SISTEMA INTEGRAL DE

RESTAURANTES-SIR, a partir de un continuo mejoramiento de los procesos ya establecidos.

Por otro lado, Julián Andrés Mera Paz, Mari Yicel Miranda Gómez, Sammy Cuaran Rosas, en su artículo nos hablan de la importancia de aplicar la norma ISO/IEC 25010 la cual enmarca la calidad en los productos software, garantizando también la articulación de los procesos para obtener los productos, por tanto, concluyeron que es una referencia óptima para la base de implementar un laboratorio de testing, lo cual era su fin de ellos.

Tomando en cuenta lo descrito anteriormente se llegó a la conclusión que para el cumplimiento de las buenas prácticas en los procesos del ciclo de vida del software es de gran importancia basarse en metodologías sistemáticas, modelos que se adecuen a las necesidades de las instituciones y estándares como la NTP ISO/IEC 12207 que ha sido las más utilizada y tomada como guía en muchas investigaciones en sus diferentes versiones, ya que brinda un modelo de procesos de referencia del ciclo de vida del software que son fundamentales para una buena ingeniería de software mejorando la productividad en el desarrollo y la calidad del producto software en cualquiera que sea el rubro de actividad de las instituciones, ya sean públicas o privadas; cabe resaltar que el uso de un marco de referencia en una entidad pública es primordial porque se rigen a normativas exigidas por el estado y manejan información sensible.

Es muy importante también que las instituciones evalúen si sus procesos están siendo ejecutados adecuadamente; es por ello que se han realizado diversas investigaciones aplicando marcos de referencia que les permitan evaluar sus procesos entre los cuales destaca la ISO IEC 25010 la cual fue el reemplazo de la ISO/IEC 9126, obteniendo como resultados el aumento de la productividad y del nivel de competitividad de las instituciones; además de la centralización y disponibilidad de su información, reducción de costos, recursos, tiempo, historial del trabajo realizado, control de avances, entre otros.

Para el desarrollo de software bajo la perspectiva documental se han encontrado tesis sobre la aplicación de metodologías que ayuden a estructurar y llevar el proceso de desarrollo del software de una manera más óptima entre ellas tenemos

2.2. Base teórica científica

Principales metodologías en la industria de software:

Una metodología de desarrollo de software, consiste principalmente en hacer uso de diversas herramientas, técnicas, métodos y modelos para el desarrollo. Regularmente este tipo de metodología, tienen la necesidad de venir documentadas, para que los programadores que estarán dentro de la planeación del proyecto, comprendan perfectamente la metodología y en algunos casos el ciclo de vida del software que se pretende seguir. Aunque actualmente existen mucha variedad en metodologías de programación. La realidad es que todas están basadas en ciertos enfoques generalistas que se crearon hace muchos años, algunos tipos de metodologías de desarrollo de software que se utilizaron e inventaron al principio de nuestra era tecnológica (OK HOSTING, Revisado Enero 2018).

A partir de eso es necesario aplicar una metodología que ahorre tiempo y mejore la calidad de los sistemas que se producen; además que sea apropiada para la documentación del desarrollo del ciclo de vida del software.

Analizando las características principales de las principales metodologías utilizadas en la industria, podemos referirnos a ellas en dos grupos diferentes: metodologías tradicionales y metodologías ágiles. Para las metodologías tradicionales tenemos los primeros modelos que ganaron popularidad por ser estructurados y centrarse principalmente en la planificación como lo son Cascada, Espiral, Prototipado, RUP (Rational Unified Procces y MSF (Microsoft Solution Framework) (Pietro Álvarez, 2015).

Respecto a las metodologías ágiles podemos definir que principalmente se busca un modelo incremental con pequeñas entregas, una menor documentación y una actitud cooperativa entre cliente y desarrollador. Dentro de estas metodologías encontramos el Modelo SCRUM, LEAN, XP (Extreme Programming), ASD (Adaptative Software Development) y Crystal Clear (Pietro Álvarez, 2015).

A continuación, se muestra una tabla comparativa sobre los dos tipos de metodologías que existen:

Tabla 1: Metodologías Ágiles vs Metodologías Tradicionales

METODOLOGIAS ÁGILES	METODOLOGIAS TRADICIONALES

METODOLOGIAS ÁGILES	METODOLOGIAS TRADICIONALES
Se basan en heurísticas (documentos de investigación o fuentes históricas) provenientes de prácticas de producción de código.	Se basan en normas provenientes de estándares seguidos por el Entorno de desarrollo.
Impuestas internamente (por el equipo).	Impuestas externamente
Pocos instrumentos de documentación de modelos. El modelado es prescindible.	Mayores instrumentos de documentación de modelos, el cual es esencial y se requiere su mantenimiento.
Actores con pocos roles, más genéricos y flexibles.	Actores con mayores roles específicos y funcionales.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
La arquitectura del software se va definiendo y mejorando.	La arquitectura se define previamente.
Se esperan cambios durante el proyecto.	Se espera que no ocurran cambios de gran impacto durante el proyecto.

Fuente: Adaptada de (Ulloa, 2014)

Los modelos tradicionales surgen para dar orden al caos en los proyectos de desarrollo de software, a través de actividades, acciones, tareas, fundamentos y productos de trabajo que permitirán llevar un mayor control y seguimiento de los mismos. Los modelos tradicionales en esencia son más estrictos o rígidos que los de desarrollo ágil, pero nos da mayor seguridad cuando se desea desarrollar un software de alta calidad (Méndez Nava , 2006).

2.3.1. Modelos de ciclo de vida de desarrollo de software

Los modelos de desarrollo de software guían la realización del proceso; aún no existe un modelo que sea considerado el definitivo, pero todos buscan que en el desarrollo de un sistema de software se maneje la complejidad, se cumpla con los requerimientos, se pueda desarrollar en tiempos razonables, su operación sea exitosa y el mantenimiento sea fácil de realizar (Medina, 2003).

Las principales diferencias entre los distintos modelos de ciclo de vida están divididas en tres grandes visiones (Cantone, 2006):

- **El alcance del ciclo de vida,** que depende de hasta donde deseamos llegar con el proyecto; solo saber si es viable el desarrollo de un producto, el desarrollo completo más las actualizaciones y el mantenimiento.
- **La cualidad y cantidad de las etapas,** en que dividiremos el ciclo de vida del software: según el ciclo de vida que adoptaremos y el proyecto para el cual lo adoptemos.
- **La estructura y la sucesión de las etapas,** si hay realimentación entre ellas y si tenemos libertad de repetirlas (iterar).

MODELOS DEL CICLO DE VIDA DEL SOFTWARE

Tabla 2: Modelos del Ciclo de Vida de Software

MODELOS ABSTRACTOS		MODELOS CONCRETOS	CONCEPTO	VENTAJAS	DESVENTAJAS
TRADICIONALES O PESADOS	**EN CASCADA**	**PURA**	Es el más básico de todos los modelos de ciclo de vida y sirve de base para otros modelos. Ve el desarrollo de un sistema como una secuencia simple de fases en donde cada fase tiene un conjunto de objetivos bien definidos y crea un resultado que sirve como insumo para la siguiente fase. Si dicho resultado es rechazado, la fase completa debe repetirse. (ver Anexo 1.1)	- La cantidad de recursos necesarios para implementar este modelo es mínima. - Debido a la estructura lógica del modelo, a menudo se pueden evitar errores conceptuales. - El modelo conduce a una extensa documentación técnica, que es un alivio para los nuevos programadores y desarrolladores y también es útil en la fase de prueba.	- Las especificaciones que se hacen inicialmente son a menudo difíciles de entender para los clientes porque son más abstractas de lo que se supone que el software debe hace. - La entrega del software lleva más tiempo porque los departamentos no trabajan simultáneamente y cada fase sólo puede comenzar cuando se ha completado la fase anterior.
		CON FASES SOLAPADAS	El modelo en cascada con fases solapadas permite hacer actividades de la siguiente fase en paralelo a las últimas actividades de la fase anterior sin romper la secuenciación de las fases y logrando una reducción del tiempo total del desarrollo del proyecto de software. (ver Anexo 1.2)	- Reducción de tiempo. - Más dinámico e integral. - Permite iterar problemáticas que surgen en el proceso. - La planificación es más sencilla.	- Más difícil de controlar el progreso del proyecto debido a que los finales de fase ya no son un punto de referencia claro. - Difícil para identificar el inicio y el final de cada etapa. - Difícil de reconocer todos los requerimientos desde un inicio.

MODELOS ABSTRACTOS		MODELOS CONCRETOS	CONCEPTO	VENTAJAS	DESVENTAJAS
		CON SUBPROYECTOS	El modelo de cascada con subproyectos, aunque divide el proyecto en subproyectos más pequeños (a partir de que se ha completado el diseño global) que se pueden desarrollar en paralelo e integrarlos todos al final, conserva el carácter secuencial de las actividades. (ver Anexo 1.3)	- Cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.	- En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso. - El proceso de creación del software tarda mucho tiempo ya que debe pasar por el proceso de prueba y hasta que el software no esté completo no se opera. Esto es la base para que funcione bien.
	EVOLUTIVOS	**ESPIRAL**	El modelo en Espiral, consiste en una serie de ciclos que se repiten, es decir que en cada fase de este modelo van a existir los mismos ciclos, con la finalidad de obtener un software, que va creciendo hasta obtener el final en el último ciclo del proceso, en esta parte tiene similitud al modelo incremental. (ver Anexo 1.4)	- Se enfoca en la eliminación de errores. - Aplica reusabilidad. - Integra desarrollo y mantenimiento. - Proporciona marco de trabajo para desarrollo de hardware y software.	- Dificultad para identificar el fin de desarrollo. - Requiere experiencia en evaluación de riesgos.

MODELOS ABSTRACTOS		MODELOS CONCRETOS	CONCEPTO	VENTAJAS	DESVENTAJAS
		ENTREGA POR ETAPAS O INCREMENTAL	el Modelo Incremental es de naturaleza interactiva, pero se diferencia de aquellos en que al final de cada incremento se entrega un producto completamente operacional. (ver Anexo 1.5)	- Con un paradigma incremental se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial. - También provee un impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del Software. - Resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos.	- El modelo Incremental no es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido, y/o de alto índice de riesgos. - Requiere de mucha planeación, tanto administrativa como técnica. - Requiere de metas claras para conocer el estado del proyecto.
		ENTREGA EVOLUTIVA O ITERATIVO	El modelo evolutivo, acepta que los requerimientos puedan cambiar en cualquier momento durante el desarrollo, en la práctica, es muy difícil que se puedan entregar todos los requerimientos al comienzo del desarrollo, en este modelo se realiza la iteración de los ciclos. (ver Anexo 1.6)	- La especificación puede desarrollarse de forma creciente. - Los usuarios y desarrolladores logran un mejor entendimiento del sistema. Esto se refleja en una mejora de la calidad del software. - Es más efectivo que el modelo de cascada, ya que cumple con las necesidades inmediatas del cliente.	- Proceso no Visible, es decir los administradores necesitan entregas para medir el progreso. - Sistemas pobremente estructurados, es decir los cambios continuos pueden ser perjudiciales para la estructura del software. - Se requieren técnicas y herramientas, es decir para el rápido desarrollo se necesitan herramientas que pueden ser incompatibles
		CASCADA EN V	El modelo en V es una variación del modelo en	- La relación entre las etapas de desarrollo y los	- Es difícil que el cliente exponga explícitamente todos

MODELOS ABSTRACTOS		MODELOS CONCRETOS	CONCEPTO	VENTAJAS	DESVENTAJAS
			cascada que muestra cómo se relacionan las actividades de prueba con el análisis y el diseño. El modelo en V hace más explícita parte de las iteraciones y repeticiones de trabajo que están ocultas en el modelo en cascada. (ver Anexo 1.7)	distintos tipos de pruebas facilitan la localización de fallos. - Es un modelo sencillo y de fácil aprendizaje. - Hace explícito parte de la iteración y trabajo que hay que revisar. - Especifica bien los roles de los distintos tipos de pruebas a realizar. - Involucra al usuario en las pruebas.	los requisitos. - El cliente debe tener paciencia pues obtendrá el producto al final del ciclo de vida. - Las pruebas pueden ser caras y, a veces, no lo suficientemente efectivas. - El producto final obtenido puede que no refleje todos los requisitos del usuario.
	HÍBRIDO	**PROCESO UNIFICADO RACIONAL**	El RUP es un modelo de proceso *Híbrido* ya que reúne elementos de modelos de procesos genéricos. Además, propone buenas prácticas para la especificación y el diseño. (ver Anexo 1.8)	- Requiere de conocimientos del proceso y de UML. - Progreso visible en las etapas tempranas. - El uso de iteraciones. - Evaluación de riesgos en lugar de descubrir en la integración final del sistema. - Facilita la reutilización del código.	- Por el grado de complejidad puede no resultar no muy adecuado - Mal aplicado en el estilo cascada.

Fuente: Elaboración propia

2.3.2 Norma ISO 12207

Esta norma presenta un modelo de procesos de referencia del ciclo de vida del software que son fundamentales para una buena ingeniería de software y cubre las mejores prácticas. El modelo de referencia es también usado para proveer una base común para diferentes modelos y métodos (Pino, Garcia, Ruiz, Piattini, 2006).

Organización de la norma ISO/IEC 12207:2017

Este estándar internacional agrupa las actividades que se pueden realizar durante el ciclo de vida de un sistema de software en cuatro grupos de procesos. Cada uno de los procesos del ciclo de vida dentro de esos grupos se describe en términos de su propósito y resultados deseados con un conjunto de actividades y tareas relacionadas que se pueden realizar para lograr esos resultados.

a) Procesos de acuerdo
b) Procesos de habilitación de proyectos organizacionales
c) Procesos de Gestión Técnica
d) Procesos técnicos

Los cuatro grupos de procesos y los procesos incluidos en cada grupo se muestran en la Ilustración 3 de la siguiente manera:

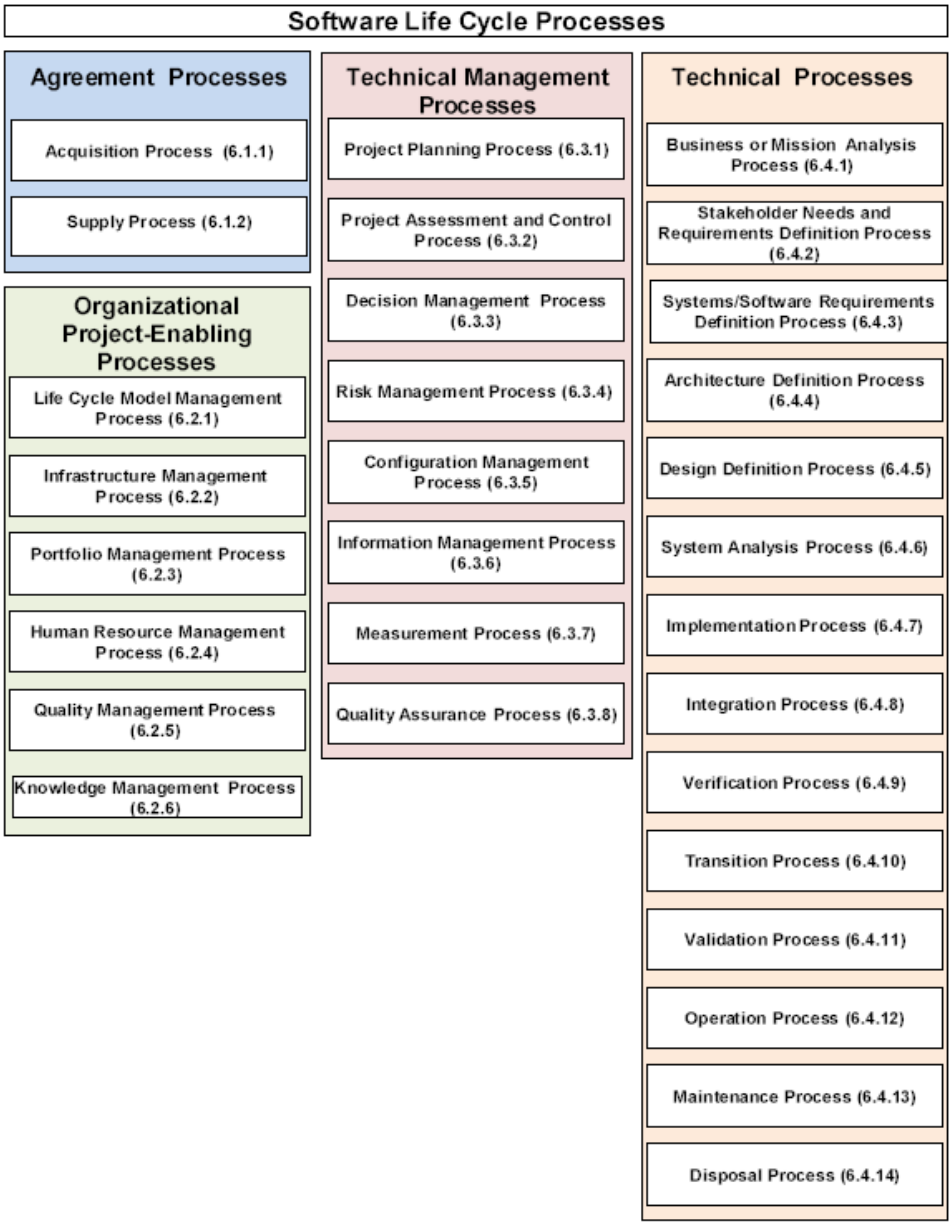


Ilustración 3: Grupos de los procesos del ciclo de vida

Fuente: ISO/IEC 12207:2017

El modelo de referencia descrito anteriormente debe ser adoptado en función de las necesidades comerciales y dominio de aplicación de las instituciones, de manera que los procesos que adopten para sus proyectos se darán de acuerdo a los requisitos de los usuarios.

Los resultados de un proceso se usan para demostrar el logro exitoso de los objetivos; esto ayuda a los evaluadores a determinar la capacidad del

proceso implementado por la institución y a proporcionar el material necesario para planificar una mejora continua.

Es por ello que a continuación se detalla la estructura de los procesos del ciclo de vida del software de la siguiente manera:

A. Procesos de acuerdo

- Proceso de adquisición
- Proceso de suministro

B. Procesos de habilitación de proyectos organizacionales

- Proceso de gestión del modelo de ciclo de vida.
- Proceso de gestión de la infraestructura.
- Proceso de gestión de cartera.
- Proceso de gestión de recursos humanos.
- Proceso de gestión de la calidad.
- Proceso de gestión del conocimiento

C. Procesos de gestión técnica

- Proceso de planificación del proyecto
- Proceso de evaluación y control del proyecto.
- Proceso de gestión de decisiones
- Proceso de gestión de riesgos.
- Proceso de gestión de la configuración.
- Proceso de gestión de la información.
- Proceso de medición
- Proceso de aseguramiento de la calidad

D. Procesos técnicos

- Análisis de negocio o misión
- Proceso de definición de necesidades y requisitos de los interesados
- Proceso de definición de requisitos del sistema / software
- Proceso de definición de arquitectura
- Proceso de definición del diseño
- Análisis del sistema
- Proceso de implementación
- Proceso de integración
- Proceso de verificación

- Proceso de transición
- Proceso de validación
- Proceso de operación
- Proceso de mantenimiento
- Proceso de eliminación

A continuación, se definirá de forma detallada las fases del modelo de cascada relacionándolas únicamente con los procesos técnicos del modelo de referencia que serán tomados en cuenta para la presente investigación.

2.3.3. Fases del ciclo de vida del software tomando como base el modelo de cascada y los procesos técnicos de la ISO 12207:2017

2.3.3.1. Fase de análisis y definición de requerimientos

La Ingeniería de Requerimientos involucra todas las actividades dedicadas a la identificación, análisis, documentación y validación de requerimientos frente a las necesidades de usuario, así como el proceso que las soporta. En ese sentido, se tomará la clasificación de actividades en dos sub disciplinas: Desarrollo de Requerimientos y Administración de Requerimientos (Karl Wiegers & Joy Beatty, 2013).

Desarrollo de Requerimientos

Antes de iniciar etapa, es necesario determinar el alcance del proyecto. Robertson y Robertson (2013) recomiendan iniciar con reuniones de despegue, que tienen como propósito principal la identificación de la funcionalidad que va a ser incluida y la que va a ser excluida del software por desarrollar.

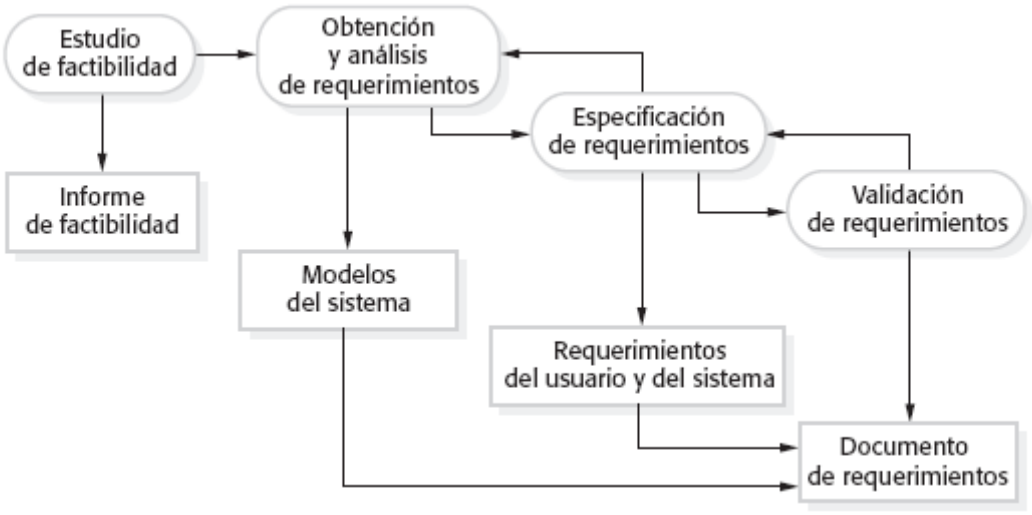


Ilustración 4: Proceso de ingeniería de requerimientos

Fuente: (Sommerville, 2011)

1) ESTUDIO DE FACTIBILIDAD

El estudio considera si el sistema propuesto tendrá un costo-beneficio y si éste puede desarrollarse dentro de las restricciones presupuestales existentes. Un estudio de factibilidad debe ser rápido y relativamente barato. El resultado debe informar la decisión respecto a si se continúa o no continúa con un análisis más detallado.

2) OBTENCIÓN Y ANÁLISIS DE REQUERIMIENTOS

Este es el proceso de derivar los requerimientos del sistema mediante observación de los sistemas existentes, discusiones con los usuarios y, análisis de tareas, etcétera. (Sommerville, 2011).

La obtención de requerimientos es la fase inicial en la cual se trata de descubrir los requerimientos e identificar los límites del sistema a través de la consulta a los participantes del sistema (stakeholders).

Es necesario para poder entender más a fondo este proceso, definir a los stakeholders o personas involucradas en el mismo. Es por ello que a continuación de muestra la siguiente tabla:

Tabla 3: Tipos de Stakehholders y funciones

PUESTO	DESCRIPCIÓN DE LAS FUNCIONES
Jefe del departamento de sistemas	Comunicar los planes, objetivos, metas, políticas, normas y procedimientos al personal a su cargo. Dirigir procesos de evaluación y cambios tecnológicos.
Jefe de proyectos	Personas que tienen a su cargo todos los aspectos del proyecto.
Usuarios	Personas u organizaciones internas o externas a la organización ejecutante que usarán el producto, servicio o resultado del proyecto.
Analista de sistemas	Analizar un sistema ya existente para comprender, mejorar, ajustar su comportamiento. Realiza el análisis para ver lo que se quiere hacer inicialmente y después darle al usuario nuevas opciones de uso.
Desarrollador	Programar los diseños propuestos por analista Capacitar sobre el uso de la tecnología Interpretar especificaciones de diseños de los sistemas. Verificar los componentes programados. Corrección de errores de compilación, ejecución.
Analista de calidad	Es el rol responsable de la planificación, diseño, implementación y evaluación de la prueba, que incluye generar el plan y el modelo de prueba, implementar los procedimientos de prueba, evaluar la envergadura y resultados de las pruebas y su efectividad.

Dentro de los objetivos de esta fase se encuentran el entender el dominio de la aplicación, las necesidades del negocio, las restricciones del sistema, a los participantes del sistema y al problema en sí, y así entender de manera inicial lo que se debe desarrollar.

Es importante empezar por definir las necesidades de las partes interesadas ya que son las describen los deseos, expectativas y limitaciones percibidas de las partes interesadas identificadas. (Zambrano, 2005)

Algunas de las técnicas y herramientas más importantes para llevar a cabo la recolección de requerimientos son:

Tabla 4: Herramientas para las diferentes fases de la ingeniería de requerimientos

HERRAMIENTA	CONCEPTO	OBTENCIÓN	ANÁLISIS	ESPECIFICACIÓN	VALIDACIÓN
Entrevistas y cuestionarios	La entrevista es un método para descubrir hechos y opiniones que tienen los posibles usuarios y otros participantes dentro del sistema que se está desarrollando.	X			
Grabaciones de video y audio	Las grabaciones suelen utilizarse en entrevistas con las partes interesadas ya que posteriormente sirven de guía al analista para revisar cada detalle que pudo haber omitido durante la entrevista.	X	X		
Brainstorming (lluvia de ideas)	Las lluvias de ideas son sesiones donde todos los participantes brindan sus ideas para obtener una solución a una problemática	X	X		
Diagramas de actividad	es un diagrama de flujo del proceso que se usa para modelar el comportamiento del sistema.		X	X	
Casos de uso	Los casos de uso describen interacciones entre los usuarios y el sistema, enfatizando en lo que el usuario necesita del sistema.	X	X	X	X

Fuente: Elaboración Propia

Estas herramientas que acabamos de ver no solo se utilizan para llevar a cabo esta primera fase de obtención de requerimientos, sino que no son restrictivas para una única actividad dentro de este gran proceso.

Una vez identificadas las necesidades de las partes interesadas deben ser priorizadas y seleccionadas porque muchas veces algunas necesidades suelen ser redundantes o ambiguas es por ello que para priorizarlas se muestra en la siguiente tabla:

Tabla 5: Categorías para la priorización de necesidades

CATEGORÍA	DESCRIPCION	EXPLICACION
Primario (P)	No debe faltar	Las necesidades son primordiales
Secundario (S)	Es útil, pero no indispensable	Las necesidades secundarias sirven de complemento a las necesidades primordiales
Opcional (O)	Es alternativo, novedoso, pero no necesario	Las necesidades son opcionales

Fuente: Elaboración propia

Finalizada esta parte, obtendremos las necesidades de las partes interesadas ya definidas para ser transformadas en los requerimientos.

Cuando ya se han establecido las necesidades como declaraciones formales, es decir ya son requerimientos, éstos para su validez tienen que cumplir con las características individuales y generales las cuales son detalladas en el análisis de requerimientos. (IEEE, Procesos del ciclo de vida - Ingeniería de Requisitos, 2011)

Los requerimientos de los interesados se analizan para las siguientes características de los requisitos individuales, así como las características del conjunto de requisitos. Las características potenciales del análisis incluyen que los requisitos son necesarios, libres de implementación, inequívocos, consistentes, completos, singulares, factibles, trazables, verificables, asequibles y acotados.

Según la ISO / IEC / IEEE 29148 (4.2.5 y 5.2.6) proporciona información adicional sobre las características de los requisitos.

En caso de que haya alguna inconsistencia, se debe acudir al stakeholder para aclararla.

Características de los requisitos individuales:

Cada requisito de los interesados, el sistema y el elemento del sistema deberá poseer las siguientes características:

Tabla 6: Descripción de características individuales que deben tener los requerimientos

Característica	Abrev.	Definición
Necesario	(N)	El requisito es actualmente aplicable y no ha quedado obsoleto por el paso del tiempo.
Implementación gratuita	(IG)	El requerimiento, al abordar lo que es necesario y suficiente en el sistema, evita colocar restricciones innecesarias en el diseño arquitectónico.
Inequívoco	(I)	El requisito se establece de tal manera que se puede interpretar de una sola manera. El requisito se establece de manera simple y es fácil de entender.
Consistente	CO)	El requisito está libre de conflictos con otros requisitos.
Completo	(COM)	El requisito establecido no necesita más amplificación porque es medible y describe suficientemente la capacidad y las características para satisfacer las necesidades de las partes interesadas.
Singular	(S)	La declaración de requisito incluye solo un requisito sin uso de conjunciones.
Factible	(F)	El requisito es técnicamente alcanzable, no requiere avances tecnológicos importantes y se ajusta a las limitaciones del sistema (por ejemplo, costo, cronograma, técnico, legal, reglamentario) con un riesgo aceptable.
Rastreable	(R)	El requisito es rastreable hacia las declaraciones de las partes interesadas documentadas específicas de la necesidad, y también se puede rastrear hacia los requisitos específicos en la especificación de requisitos.
Verificable	(V)	El requisito tiene los medios para demostrar que el sistema satisface el requisito especificado.

Fuente: (IEEE, Procesos del ciclo de vida - Ingeniería de Requisitos, 2011):

Características de un conjunto de requisitos:

Hay ciertas características que deben tenerse en cuenta para el conjunto de requisitos de los elementos del sistema de partes interesadas. Esto asegurará

que el conjunto de requisitos proporcione colectivamente una solución factible que satisfaga las intenciones y limitaciones de las partes interesadas.

Cada conjunto de requisitos deberá poseer las siguientes características

Tabla 7: Descripción de características que debe tener un conjunto de requerimientos

Característica	Abrev.	Definición
Completo	(COM)	El conjunto de requisitos no necesita más amplificación porque contiene todo lo pertinente a la definición del elemento del sistema o sistema que se especifica.
Consistente	(CON)	El conjunto de requisitos no tiene requisitos individuales que sean contradictorios. Los requisitos no están duplicados
Asequible	(ASE)	El conjunto completo de requisitos puede satisfacerse mediante una solución que sea obtenible / factible dentro de las restricciones del ciclo de vida (por ejemplo, costo, cronograma, técnico, legal, reglamentario).
Encerrado	(ENC)	El conjunto de requisitos mantiene el alcance identificado para la solución prevista sin aumentar más allá de lo que se necesita para satisfacer las necesidades del usuario.

Fuente: (IEEE, Procesos del ciclo de vida - Ingeniería de Requisitos, 2011).

Después de la obtención y análisis de requerimientos se puede incluir modelos de sistemas, lo que ayuda a entender el sistema que se va a especificar.

Los modelos de sistemas son representaciones del modelo de negocio de la concepción del requerimiento de software. Los modelos que detallaremos a continuación son representaciones del modelo funcional y de datos de los sistemas; dependiendo del ciclo de vida o metodología para el desarrollo, se deberá definir el modelo más apropiado para utilizar. (Durand, 2017)

Tipos de modelos:

a. Modelo de contexto

Los modelos de contexto son utilizados para las etapas tempranas del relevamiento de información, en los modelos de proceso de negocio; esto, para conocer la funcionalidad y la descripción del requerimiento de usuario a alto nivel.

b. Modelado para el desarrollo estructurado

Se enfoca en la descomposición funcional del sistema, su objetivo es obtener una definición completa del software por desarrollar, a través de la elicitación de requisitos. Asimismo, se definen los datos de entrada y salida. Los diagramas utilizados en este modelo son:

- Diagrama de flujo de datos
- Diagrama entidad–relación
- Diagrama de transición de estados

El diccionario de datos es un diagrama complementario que permite conocer el detalle de los datos.

c. Modelado para el desarrollo orientado a objetos

En este modelo, la unidad básica de construcción es la clase y el objeto; el modelo muestra los datos del sistema, así como su procesamiento.

Si bien es cierto existen herramientas; pero entre ellas la más utilizada es el Lenguaje Unificado de Modelado (UML).

UML está respaldado por el OMG (Object Management Group), encargado del mantenimiento de estándares y nuevas versiones del lenguaje unificado. Proporciona diagramas que esquematizan los requisitos funcionales en términos técnicos que facilitan el desarrollo del software.

Se recomienda trabajar con UML porque se puede utilizar para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Es un lenguaje muy expresivo, que cubre todas las vistas necesarias para desarrollar y luego desplegar tales sistemas.

3) ESPECIFICACIÓN DE REQUERIMIENTOS

Consiste en la actividad de transcribir la información recopilada durante la actividad de análisis, en un documento que define un conjunto de requerimientos. En este documento se incluyen dos clases de requerimientos.

Los requerimientos del usuario son informes abstractos de requerimientos del sistema para el usuario final del sistema; y los requerimientos de sistema son una descripción detallada de la funcionalidad a ofrecer. (Sommerville, 2011)

Tipo de requerimientos**:**

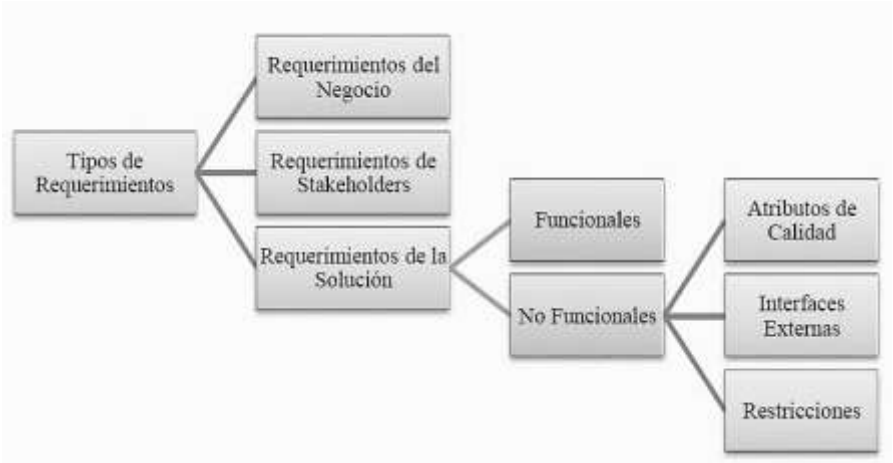


Ilustración 5: Tipos de Requerimientos
Fuente: (Lozada, 2017)

Requerimientos del Negocio:

Son aquellos requerimientos que representan los objetivos establecidos por la organización. Son la base principal para el desarrollo del proyecto porque describen las necesidades que el Software cubrirá sin descuidar las reglas de negocio de la organización. (José Martinez Guerero, Camilo Silva Delgado, 2010)

Requerimientos de usuario:

Otra cosa importante es que solo se manejará el documento de requerimientos para ellos, ya que solo se debe describir el comportamiento del sistema (José Martinez Guerero, Camilo Silva Delgado, 2010).

Requisitos del sistema:

Los requerimientos del sistema son descripciones más detalladas de las funciones, los servicios y las restricciones operacionales del sistema de software. El documento de requerimientos del sistema (llamado en ocasiones especificación funcional) tiene que definir con exactitud lo que se implementará. (Sommerville, 2011)

Requisitos funcionales:

Describen el comportamiento y los datos que el sistema administrará. También, las capacidades que el sistema podrá realizar en términos de comportamientos o acciones o respuestas de aplicación de tecnología de la información específicas de las operaciones (Lozada, 2017).

Sobre la base de los requerimientos funcionales se generan los casos de uso del sistema. Debe existir una correspondencia entre ambos.

La finalidad de esta actividad es especificar cada caso de uso desarrollando los escenarios, y describiendo los flujos principales y los alternativos. Se deberá generar como parte de esta actividad lo siguiente (Durand, 2017):

- Diagramas de casos de uso
- Diagrama de actividades
- Prototipos del sistema

Requisitos no funcionales:

Este tipo de requisitos están relacionados con la operación del software y hacen referencia al comportamiento de la solución, describen condiciones ambientales bajo las cuales la solución debe permanecer activa por determinados períodos de tiempo o cualidades que los sistemas deben tener a nivel operacional. (Durand, 2017).

A continuación, mostraremos la estructura de los requisitos no funcionales:

- Atributos de calidad:

  Son las propiedades de calidad de un sistema, tales como: confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad.

Tabla 8: Propiedades de calidad de un sistema

Comportamiento	Descripción
Funcionalidad	La capacidad del sistema para proveer las funciones que cumplan las necesidades de los usuarios, es decir, que cumplan con los requerimientos del negocio y de usuarios.
Fiabilidad	La capacidad del sistema para mantener su nivel de rendimiento en determinados periodos de tiempo establecidos por la organización.
Usabilidad	La capacidad del sistema para ser fácilmente entendido, aprendido y de uso sencillo para el usuario final.
Eficiencia	La capacidad del sistema para lograr el rendimiento deseado, teniendo en cuenta recursos físicos utilizados y condiciones específicas de uso. También tiene en cuenta los tiempos de respuesta de la solución.
Mantenibilidad	La capacidad que tiene el sistema para ser actualizado, modificado o corregido según sea el caso.
Portabilidad	La capacidad que tiene el sistema para ser desplazado entre diferentes entornos, lo que representa adaptabilidad al medio en donde sea colocado y su forma de interactuar con los demás sistemas.

Fuente: (José Martinez Guerero, Camilo Silva Delgado, 2010)

- Requerimientos interfaces externas:

  Describen las conexiones entre un sistema de software, un usuario, otro sistema o un dispositivo de hardware (Lozada, 2017).

- Restricciones:

  Las restricciones técnicas están relacionadas con las limitaciones de las plataformas y las arquitecturas de hardware y software que se utilizan dentro de un proyecto. Es debido a estas limitaciones que se pueden presentar dependencias entre elementos de una infraestructura tecnológica. (Zambrano, 2005)

  Algunas restricciones pueden estar en las fechas de entrega, especificaciones de hardware, limitaciones, especificaciones de software, colores de preferencia, uso de gráficos y logos, entre otros.

La elaboración de las especificaciones de requerimientos de usuario y de software depende de las políticas y necesidades de la organización y/o del proyecto; en algunos casos las dos especificaciones se combinan en un solo documento, o incluso se omite la elaboración de la ERS.

En la actividad de especificación de requerimientos se deben realizar las siguientes tareas (Murali, 2013):

- Evaluar factibilidad de cada requerimiento**:**

Mediante una reunión con los stakeholders, evalúan la factibilidad técnica, económica y de tiempo de cada requerimiento: (Lozada, 2017)

Técnica: Se refiere a que la implementación del requerimiento no está limitada por hardware, software, el estado actual de la tecnología o el estado actual de la organización en tecnología.

Económica: Se refiere al que la implementación del requerimiento no está limitada por el presupuesto.

De tiempo: Se refiere a que la implementación del requerimiento es factible dentro del tiempo disponible.

La participación los stakeholders es importante ya que es necesario tomar decisiones en el caso de que un requerimiento no sea factible, lo que se deberá dejar constancia en el Acta de Reunión.

- Priorizar los requerimientos**:**

Colocar el orden en el cual se van a implementar los requerimientos de acuerdo a la necesidad del stakeholder. El orden se determinará mediante la asignación de prioridad Alta, Media o Baja (García A. P., 2013).
Este mecanismo es llevado a cabo para medir el grado de importancia de los requerimientos.

De esta manera se determinó que cada requerimiento se debía clasificar dentro de las siguientes prioridades:

Tabla 9: Cuadro de prioridades de un requerimiento

Prioridad	Descripción
Alta	Es un requerimiento crítico que debe incluirse en versiones tempranas del producto. Si no se implementa puede afectar la satisfacción del usuario.
Media	El no incluir este tipo de requerimiento puede afectar la satisfacción del usuario, pero no se debe retrasar las primeras versiones del producto por la ausencia de uno de ellos.
Baja	Si no se incluye en las primeras versiones del producto no se espera un impacto significante en la satisfacción del usuario.

Fuente: (García A. P., 2013).

- Mecanismo de control de estado de los requerimientos:

Con el fin de mantener un control del estado de los requerimientos funcionales y no funcionales, se definen los siguientes estados que un requerimiento puede tener a lo largo del proyecto:

Tabla 10: Estados de los requerimientos

Estado	Descripción
Especificado	Ha sido especificado en la plantilla de requerimientos.
Diseñado	El requerimiento tiene algún elemento de diseño asociado.
Implementado	Ha sido desarrollado, pero no se ha sometido a un proceso de pruebas.
Aprobado	El requerimiento pasó satisfactoriamente el proceso de pruebas.

Fuente: (García A. P., 2013)

4) VALIDACIÓN DE REQUERIMIENTOS

Esta actividad verifica que los requerimientos sean realistas, coherentes y completos. Durante este proceso es inevitable descubrir errores en el documento de requerimientos. En consecuencia, deberían modificarse con la finalidad de corregir dichos problemas. (Sommerville, 2011)

Para cumplir con la validación y verificación de requerimientos, se utilizan técnicas de revisión informal e inspecciones. En las revisiones informales se invita a uno o varios compañeros de trabajo para la revisión de la especificación de requerimientos, solicitando sus comentarios al respecto; con el objetivo de producir un reporte de los problemas detectados en los requerimientos.

Se muestran las tareas necesarias para cumplir una inspección de requerimientos de software, según Wiegers y Beatty (2013).

Tabla 11: Proceso de inspección de requerimientos de software

Tarea	Descripción
Planeación	Determinar cuándo y dónde va a ser la reunión, quién va a participar, el material que se va a revisar y el tiempo requerido.
Preparación	Previo a la reunión, entregar el material a los participantes para su revisión.
Reunión de inspección	Realizar la reunión de inspección en la que se deberá: 1) Leer cada requerimiento, 2) Solicitar a los participantes que indiquen los problemas y errores encontrados y 3) Registrarlos.
Corrección	Resolver los problemas y corregir los errores encontrados en los requerimientos.
Verificación	Verificar que todos los problemas identificados han sido resueltos y los errores han sido corregidos.

Fuente: (Lozada, 2017).

Administración de requerimientos

Una vez que los requerimientos han sido identificados, documentados y validados, se debe controlar su evolución a lo largo del ciclo de vida del desarrollo de software. En ese sentido, es necesario establecer un conjunto de actividades que permitan gestionarlos.

Estas actividades son: Control de cambios, Control de versiones, Trazabilidad de requerimientos y Seguimiento a requerimientos (Karl Wiegers & Joy Beatty, 2013).

Ilustración 6: Proceso de administración de requerimientos
Fuente: (Lozada, 2017)

CONTROL DE VERSIONES

Luego de que los requerimientos han atravesado las actividades de validación, verificación y aprobación, pasan a formar parte de una línea base.

El desarrollo de una línea de base crea una referencia a utilizar para realizar un seguimiento de las necesidades que evolucionan con el tiempo.

Cada cambio aprobado modificará la versión de un requerimiento, de un conjunto de requerimientos o de otros artefactos del proyecto. (Cueva, 2014)

Esto implica cambios en el código fuente es por ello que existen sistemas de control de versiones, los cuales son herramientas que gestionan los diversos cambios que se realizan sobre los elementos de uno o más proyectos, guardando así versiones de nuestra aplicación en todas sus fases de desarrollo. (Anexo 25 – Obs. 4)

Existen 3 tipos de sistemas de control de versiones los cuales con Sistema de control de versiones local, centralizado y distribuido y cada uno posee ventajas y desventajas las cuales se muestran en el Anexo 3.

CONTROL DE CAMBIOS

A lo largo del ciclo de vida de desarrollo del software los requerimientos pueden modificarse, eliminarse o añadirse debido a diferentes factores, es por eso que se deben establecer mecanismos y políticas para reconocer, evaluar y decidir como integrar las nuevas necesidades e ir evolucionando hacia una línea de base de las necesidades existentes. (Cueva, 2014)

Ilustración 7: Diagrama de transición de estados para el ciclo de vida de las peticiones de cambio
Fuente: (Lozada, 2017)

Todas las peticiones de cambio deben ser analizadas para evaluar el impacto de su implementación en el sistema.

Estos aspectos se llevan a cabo en el análisis de requerimientos, pero debido a que existe una nueva petición de cambio se tiene que realizar un nuevo análisis de este requerimiento. (Anexo 23 – Obs. 1)

Para una correcta identificación de los controles de cambios de los requerimientos de las organizaciones de desarrollo de software, se identifican las siguientes actividades:

- Valorar el cambio: Otro punto importante es valorar la factibilidad de la solicitud realizada ya sea por un cliente interno o uno externo. Para ello se deberá ir recorriendo todo el árbol de requisitos viendo cómo les afecta el cambio, y aquí es donde entra la trazabilidad de los requisitos.

- Analizar Modificación: El líder de implementación debe realizar el análisis de la solicitud para saber que tanto impacta la modificación e identificar puntualmente las modificaciones solicitadas que afectan el requerimiento completo y así identificar si el cambio afecta más de un requerimiento.

- Documentar Cambio: Para tener un mejor control sobre los cambios solicitados es recomendable realizar una documentación clara para evitar ambigüedades en las modificaciones que se van a realizar a los requerimientos.

- Aprobación Control de Cambios Aprobar Cambios: Una vez se ha analizado el impacto del cambio, se debe tomar una decisión. Si se acepta el cambio, tras negociarlo con el cliente, se continuará con la actividad de implementar el cambio. En caso contrario, se deberá negociar con el cliente el siguiente paso a realizar.

- Planear Cambio: Después de tener una aprobación formal del cambio aceptado se planea el tiempo necesario y los recursos necesarios para llevar a cabo el cambio aprobado.

- Realizar Cambio: Una vez se planea el cambio aprobado se debe realizar las modificaciones necesarias a todos los productos que resulten afectados por dicho cambio.

- Revisar Cambio: Una vez se realice el cambio es recomendable hacer una verificación por parte del líder para identificar que el requerimiento incluye todos los cambios solicitados y que fueron aprobados.

- Actualizar Línea Base: Es recomendable utilizar el nuevo requerimiento como línea base, esto con el fin de trabajar siempre sobre la última versión del requerimiento. Informar: Una vez se realice la modificación de la solicitud se debe informar a los interesados que el cambio ya está realizado para que sea verificado por el cliente.

En realidad, el impacto se relaciona directamente con la fase de desarrollo de software en la cual se recibe la petición de cambio. En la tabla se muestran los artefactos del proyecto.

Tabla 12: Artefactos impactados y estrategias de implementación de una petición de cambio

Fase	Artefactos impactados
Fase de requerimientos de usuario y del sistema	Documento de requerimientos de usuario y del sistema.
Fase de diseño	Documento de requerimientos de usuario y del sistema y documento de diseño.
Fase de desarrollo	Documento de requerimientos de usuario y del sistema y documento de diseño

Fase	Artefactos impactados
	código fuente.

Fuente: Elaboración propia

TRAZABILIDAD DE REQUERIMIENTOS

La trazabilidad es el seguimiento que se pueda hacer sobre el requerimiento, estableciendo relaciones entre el requerimiento son su origen y los artefactos que se generen al lograse haber implementado, es esencial registrar trazabilidad y dar seguimiento a cada requerimiento, con el objetivo de garantizar que todas las necesidades del usuario sean atendidas. (José Martinez Guerero, Camilo Silva Delgado, 2010)

Tabla 13: Descripción de las tareas para crear las matrices de trazabilidad

N°	Tarea	Descripción
1	Crear trazabilidad de requerimientos funcionales a elementos de diseño.	Crear la relación de trazabilidad cuando se ha completado el diseño de software para cualquier módulo.
2	Crear trazabilidad de requerimientos funcionales a código fuente.	Crear la relación de trazabilidad cuando se ha finalizado el código para cualquier componente.
3	Crear trazabilidad de requerimientos funcionales a casos de prueba.	Crear la relación de trazabilidad cuando se ha finalizado las pruebas unitarias o de integración de un componente o módulo.
4	Crear trazabilidad de peticiones de cambio a requerimientos funcionales.	Crear la relación de trazabilidad cuando una petición de cambio ha sido implementada.
5	Controlar la información de trazabilidad periódicamente.	Controlar el registro de la trazabilidad para asegurar que todos los requerimientos han sido cubiertos.

Fuente: (Lozada, 2017)

SEGUIMIENTO A REQUERIMIENTOS

El seguimiento a los requerimientos se da mediante la identificación y documentación de cómo los requisitos están relacionados de forma lógica e intenta asegurar que los requerimientos han sido implementados de forma completa. (Cueva, 2014)

DOCUMENTACIÓN

El propósito de la documentación de requerimientos es el comunicar los requerimientos a los diferentes participantes del desarrollo de software. El documento de requerimientos de software es una herramienta que puede servir como base para la evaluación de los requerimientos en el producto, y para la evaluación misma del proceso que se llevó a cabo (evaluar las actividades de diseño, implementación, las pruebas realizadas y la verificación y validación de estas actividades. (Zambrano, 2005)

2.3.3.2. Fase de Diseño y Análisis del sistema

A. Análisis del sistema

Dado que la arquitectura describe la estructura de todo el sistema, es posible realizar un análisis en etapas tempranas del desarrollo permitiendo localizar los posibles errores de diseño y así tomar decisiones de diseño. En este análisis también se evalúa si el sistema puede cumplir con los requerimientos no funcionales como son rendimiento, escalabilidad, contabilidad, entre otros.

El objetivo principal del Análisis es plasmar todo el avance en el documento de análisis del sistema (DAS), que es fundamentalmente un documento de carácter técnico. El DAS debe contener principalmente modelos del sistema software a desarrollar. La realización de estos modelos permite a los analistas identificar problemas en los requisitos y dar el primer paso hacia la implementación del sistema.

Estructura y dependencias del Análisis del Sistema

El DAS puede considerarse como un complemento a la Especificación de Requisitos del Sistema (ERS), ya que contiene la arquitectura lógica, los modelos conceptuales y la especificación de las interfaces del sistema software especificado mediante los requisitos de la ERS, a los que debe estar convenientemente trazado, tal como se muestra en la siguiente figura.

Ilustración 8: Relación del Documento de Análisis del Sistema con la Especificación de Requisitos del Sistema
Fuente:(ANDALUCÍA, 2013)

Esto involucra el análisis de las necesidades del sistema. Existen herramientas y técnicas especiales que ayudan al analista a realizar las determinaciones de los requerimientos. Las herramientas como los diagramas de flujo de datos (DFD) para graficar la entrada, los procesos y la salida de las funciones de la empresa, o los diagramas de actividad o de secuencia para mostrar la secuencia de los eventos, sirven para ilustrar a los sistemas de una manera estructurada y gráfica. A partir de los diagramas de flujo de datos, de secuencia u otros tipos de diagramas se debe desarrollar un diccionario de datos para enlistar todos los elementos de datos utilizados en el sistema, así como sus especificaciones. (Kenneth E. Kendall Y Julie E. Kendall, 2011)

B. Definición arquitectura

Uno de los factores que determina el éxito o fracaso de un sistema de software, es su arquitectura. Con esto nos referimos a la estructura o conjunto de estructuras del sistema, las cuales están compuestas de elementos de software, propiedades externas visibles de estos elementos y las relaciones entre sí.

La arquitectura de software se encarga de relacionar y negociar entre el diseño y la implementación de alto nivel de la estructura del software.

Es por ello que, para representar la arquitectura de un sistema de software, se utilizara un lenguaje de descripción de arquitectura de tal manera que sea entendible por la mayoría de las personas que están involucradas en el desarrollo.

El lenguaje de descripción de arquitectura que se usará es del lenguaje de modelado unificado (UML), este es un lenguaje estándar que nos permite

modelar los componentes de un sistema de forma que sea entendible (Mora, 2011).

Este enfoque utiliza diferentes vistas para separar los conceptos de cada stakeholder. El enfoque de 4+1 vistas ha sido ampliamente por la comunidad de la industria de software para representar el diseño de la arquitectura de software de las aplicaciones.

En la Ilustración 8 se muestra los diagramas de UML que corresponden a cada vista del modelo de 4+1 vistas.

Existen diferentes opiniones relativas a qué vistas se requieren. El conocido modelo de vista 4+1 de la arquitectura de software, sugiere que deben existir cuatro vistas arquitectónicas fundamentales, que se relacionan usando casos de uso o escenarios.

Las vistas que él sugiere son (Kruchten, 1995):

Ilustración 9: Vistas de Arquitectura
Fuente: (Kruchten, 1995)

- **Vista lógica**

  Apoya principalmente los requisitos funcionales, lo que el sistema debe brindar en términos de servicios a sus usuarios (Kruchten, 1995).

  El sistema se descompone en una serie de abstracciones primarias, tomadas principalmente del dominio del problema en la forma de objetos o clases de objetos (Landeta.P, 2013).

Diagramas: diagramas de clases, diagramas de estado, diagrama de colaboración.

a. Diagrama de clases

Es un diagrama estático que permite visualizar las clases, sus atributos, sus comportamientos(métodos) y las relaciones entre ellos, describiendo así la estructura lógica del sistema. (ICESI, 2010).

b. Diagrama de colaboración

Este tipo de diagrama muestra las interacciones que ocurren entre los objetos que participan en una situación determinada (Maldonado, 2015).

- **Vista de procesos**

Se tratan los aspectos de concurrencia y distribución, integridad del sistema y tolerancia a fallos.

Se especifica en cual hilo de control se ejecuta efectivamente una operación de una case identificada en la vista lógica.

Diagramas: diagrama de actividad, diagrama de estado, diagrama de secuencia

a. Diagrama de actividad

Un diagrama de actividades ilustra la naturaleza dinámica de un sistema mediante el modelado del flujo ocurrente de actividad en actividad. Una actividad representa una operación en alguna clase del sistema y que resulta en un cambio en el estado del sistema.

b. Diagrama de estado

El diagrama de estados engloba todos los mensajes que un objeto puede enviar o recibir, en otras palabras, es un escenario que representa un camino dentro de un diagrama. (Shirley, 2012)

c. Diagrama de secuencia

El diagrama de secuencia muestra el detalle del escenario a nivel de actividades. diferencia del caso de uso, que muestra el escenario del flujo principal y escenarios alternativos del sistema. (Durand, 2017)

- **Vista de desarrollo**

  Se centra en la organización real de los módulos de software en el ambiente de desarrollo.

  Diagrama: diagrama de componentes

  a) Diagrama de componentes

  Los componentes pertenecen a un mundo físico, es decir, describen la organización de los componentes físicos de un sistema.

  Cada componente debe tener un nombre que lo distinga de los demás. (Shirley, 2012)

- **Vista física**

  Se toma en cuenta los requisitos no funcionales del sistema tales como, disponibilidad, confiabilidad, desempeño entre otras más.

  Diagrama: diagrama de despliegue

  i) Diagrama de despliegue

  El diagrama de despliegue permite mostrar la arquitectura en tiempo de ejecución del sistema respecto al hardware y software. (Pressman, 2010).

- **Vista de escenarios**

  La función de los escenarios es relacionar las 4 vistas entre sí, de esta forma se cuenta una perspectiva general del sistema, que ayuda a descubrir nuevos elementos o validar la arquitectura (Landeta.P, 2013).

  Diagrama de caso de uso

  Estos diagramas muestran operaciones que se esperan de una aplicación o sistema y como se relaciona con su entorno, es por ello que se ve desde el punto de vista del usuario. (Shirley, 2012)

  A lo largo de la explicación de cada una de las vistas su translación a un lenguaje de modelado concreto como UML. Hay que tener en cuenta que UML nace casi a la vez que el modelo 4+1, por lo que en un origen no existía una clara relación entre ambos, lo que a menudo produce confusión al diseñador que en la actualidad quiere modelar una arquitectura con

ambas herramientas. A modo de resumen la translación se presenta en la siguiente tabla. (Adolfo, 2014)

Tabla 14: Relación entre vistas de arquitectura y UML

Vista	UML
Escenarios	Casos de Uso
Lógica	Clases, de Estados y Colaboración
Desarrollo	Componentes
Física	Despliegue
Procesos	Actividad, Estados, Secuencia

Fuente: (Adolfo, 2014)

- **Definición de diseño**

  En esta etapa se describe la estructura, los componentes, los conectores, las interfaces y los algoritmos utilizados en el software que se desea implementar. La etapa de diseño se desarrolla de forma iterativa a través de varias versiones, rara vez los diseñadores llegan a un diseño detallado inmediatamente.

  Modelo general del proceso de diseño

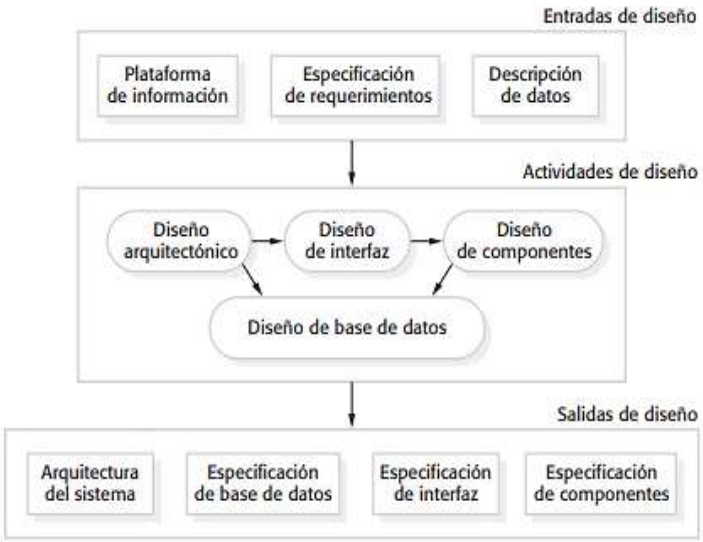


Ilustración 10: Proceso de Diseño del Software
Fuente: (Sommerville, 2011)

Las actividades en el proceso de diseño varían dependiendo del tipo de sistema a desarrollar. La figura muestra cuatro actividades que podrían formar parte del proceso de diseño para sistemas de información:

- **Diseño arquitectónico**

  Aquí se identifica la estructura global del sistema, los principales componentes (llamados en ocasiones subsistemas o módulos), sus relaciones y cómo se distribuyen (Sommerville, 2011). Para ello es necesario el buen entendimiento de los requerimientos que se plasman en la primera fase.

  A la hora de diseñar la arquitectura de cualquier software, el desarrollador no parte de cero, existen unos modelos o patrones de diseño en los que se basan la mayoría de sistemas. A continuación, se muestra un cuadro comparativo con algunos patrones de arquitectura. (Anexo 24 – Obs.3)

Tabla 15: Cuadro comparativo de patrones de arquitectura

Patrones de arquitectura	Definición	Ventajas	Desventajas
Arquitectura en capas	La arquitectura basada en capas se enfoca en un estilo de programación en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño.	Si se despliega en diferentes servidores, permite hacer al software más escalable, más seguro y mayor disponibilidad y usabilidad.	Un cambio en la funcionalidad podría implicar modificaciones en varias capas. Trabajo innecesario por parte de capas más internas o redundante entre varias capas
Modelo Vista presentador	Es un patrón para el desarrollo de interface de usuario que se caracteriza por trabajar con una clase y dos interfaces. La primera clase es llamada presentador que actúa como intermediario entre la vista (como interface) y el modelo (como interface).	• simplificar las tareas de desarrollo y mantenimiento del software escrito con estos a través de la división de ocupaciones. • La vista puede ser fácilmente cambiada • La vista puede ser trabajada incluso por una persona diferente	Requiere una gran cantidad de código que no vale la pena hacer en proyectos fáciles y simples. Requiere que crees interfaces para que puedas usar el patrón observador por cada acción
Modelo Vista Controlador	Este patrón nos separa el modelo de datos, el modelo de la capa de presentación (vista) y de la parte de control.	Permite que los datos cambien de manera independiente de su representación y viceversa. Mejora la arquitectura y la robustez, fuerza a utilizar una arquitectura modular, fomenta la separación entre capas, favorece la reutilización.	Puede implicar código adicional y complejidad de código cuando el modelo y las interacciones son simples.

Patrones de arquitectura	Definición	Ventajas	Desventajas
CQRS (Command Query Responsability Sgregation)	es un patrón arquitectónico que se divide en dos modelos diferentes; el modelo de consultas para la lectura de datos y el modelo de comandos para la ejecución de transacciones.	Funciona bien para dominios complejos y provee separación de Preocupaciones. La seguridad mejora previniendo el acceso de lectura o escritura no deseados.	Si se requieren operaciones CRUD básicas se agrega mucha complejidad innecesaria.
Modelo vista vista modelo	En gran parte basado en el patrón Modelo Vista Controlador (MVC), MVVM está dirigido a modernas plataformas de desarrollo de interfaz de usuario que soportan programación orientada a eventos, como HTML5, Windows Presentation Foundation WPF, Silverlight y el Framework ZK.	tiene por objetivo simplificar las tareas de desarrollo y mantenimiento del software escrito con estos a través de la división de ocupaciones. Mayor claridad y mejor comprensión del proyecto frente a otros desarrolladores.	Muy poco acoplamiento con el componente y con la vista
Arquitectura orientada a servicios	Esta arquitectura permite crear sistemas altamente escalables, que pueden ayudar a las organizaciones a impulsar el rendimiento y, al mismo tiempo, reducir costos de TI y mejorar la flexibilidad en los procesos del negocio	Mejora en los tiempos de realización de cambios en procesos. Facilidad para evolucionar a nuevos modelos de negocios	requiere un cambio en las organizaciones, un alto esfuerzo. No siendo sencillo, para la mayoría de las organizaciones adoptar SOA.

Fuente: Elaboración propia

Después de haber revisado los documentos sobre arquitecturas de software y los diferentes modelos que existen, el analista debe tomar la decisión de que modelo es el más adecuado de acuerdo a las características del software que se desea desarrollar.

El diseñar debidamente una arquitectura de software, garantiza que el sistema de software cumpla con uno o varios atributos de calidad. Los atributos de calidad se clasifican en dos grupos: operacionales y de desarrollo; los atributos operacionales son las cualidades del sistema que están en operación, por ejemplo: rendimiento, confiabilidad, disponibilidad, tolerancia a fallas. En cambio, los atributos de desarrollo son las cualidades del sistema que son relevantes desde una perspectiva del desarrollo de software, por ejemplo: facilidad de modificación, facilidad de re-utilización, flexibilidad. (Jan Bosch & Peter Molin, 1999)

Si la arquitectura no se diseña de forma apropiada, el sistema de software resultante no logrará sus objetivos; es por ello que existen varias técnicas para evaluar arquitecturas de software, que se clasifican en cualitativas y

cuantitativas. Dentro de las técnicas de evaluación cualitativas se pueden utilizar: escenarios, cuestionarios o listas de verificación. (ejemplo: véase el Anexo 3) Por otro lado, en las técnicas de evaluación cuantitativas se pueden emplear: métricas, simulaciones, prototipos, modelos matemáticos. Para ello véase el Anexo 2 el cual muestra tablas de métricas para evaluar los atributos de calidad. (Yan Liu & Ian Gorton, 2005)

- **Diseño de interfaz**

  En éste se definen las interfaces entre los componentes de sistemas; con una interfaz precisa, es factible usar un componente sin que otros tengan que saber cómo se implementó. Una vez que se acuerdan las especificaciones de interfaz, los componentes se diseñan y se desarrollan de manera concurrente. (Sommerville, 2011).

  Una vez identificadas las tareas del usuario, se crean y analizan los escenarios para éste y se define un conjunto de objetos y acciones de la interfaz. (Pressman, 2010).

  Criterios para una visualización de interfaz atractiva:

Tabla 16: Criterios para una óptima visualización de interfaz

Versión Desktop	Versión Web
Tamaño de ventanas y controles, como botones, menús desplegables, cuadrículas y enumerar las vistas de manera adecuada en función del contenido esperado.	Poner énfasis en la legibilidad, incluido el uso de encabezados, espaciado adecuado, fuentes legibles, colores atractivos
Si se hace que una ventana sea redimensionable y el contenido se trunca, asegurarse de que se muestre más contenido a medida que la ventana se hace más grande	Mantenga el contenido conciso evitando el texto excesivo.
Etiquetar cada control y cada grupo de controles y asegurarse de que el orden de tabulación de los controles en una ventana sea correcto.	Asegurarse de que la navegación y los menús no sean demasiado complicados que eviten que los usuarios encuentren lo que quieren
Ser consistente a lo largo de la aplicación con cosas como colores, íconos, fuentes y subtítulos/términos	Sea consistente a lo largo de la aplicación con cosas como colores, íconos, fuentes y subtítulos/términos.
La aplicación debe mostrar mensajes informativos para el usuario, como mensajes de confirmación, advertencia y error.	Mostrar indicadores de progreso cuando el usuario tiene que esperar una respuesta, como una animación de espera o un porcentaje completado indicador

Fuente: Elaboración propia

Elementos del diseño:

Para elaborar el diseño de la interfaz se utilizarán las siguientes herramientas (ITCA, s.f.):

- Diagrama de menús
- Diseño de cada una de las pantallas del sistema de acuerdo con el diagrama jerárquico.

a. Esquemas de menú

Los menús proporcionan a los usuarios una forma sencilla y familiar de recuperar información e interactuar con el sistema. A los usuarios se les presenta el diseño completo del sistema donde se muestran las distintas opciones que el sistema tendrá. En el esquema de menús, es importante dejar claros los diferentes subsistemas que contendrá el sistema, cada opción importante que le corresponde. (Aranda, 2019)

Además de ello, es importante reflejar el tipo de información que se podrá consultar o los reportes que se pueden generar.

b. Diseño de pantallas

Las formas se utilizan como una herramienta para hacer que el trabajo se realice e introducir datos al sistema. Al hacer un análisis de formas, se debe determinar qué datos se trata de capturar. Las formas en papel siguen siendo los vehículos principales empleados para introducir los datos al sistema. (Aranda, 2019)

Lineamientos para el diseño de formularios:

- Haga que las formas sean fáciles de llenar.
- Asegúrese de que las formas satisfacen el objetivo para el que fueron diseñadas.
- Diseñe formas que aseguren el llenado preciso.
- Las formas deben ser atractivas.

Documentos de control

- Prototipos de pantalla

- **Diseño de componentes**

  En él se toma cada componente del sistema y se diseña cómo funcionará. Esto puede ser un simple dato de la funcionalidad que se espera implementar, y al programador se le deja el diseño específico. (Sommerville, 2011)

  El objetivo es traducir el modelo del diseño a software operativo. La traducción es difícil y está abierta a la introducción de errores sutiles que son difíciles de detectar y de corregir en las etapas posteriores del proceso del software. (Pressman, 2010)

  Documentos de control
  - Diagramas de Clase
  - Diagramas de despliegue
  - Diagrama de componentes

- **Diseño de base de datos**

  Donde se diseñan las estructuras del sistema de datos y cómo se representarán en una base de datos. El trabajo aquí depende de si una base de datos se reutilizará o se creará una nueva.

  - Diagrama de Modelo Entidad Relación

  a. El modelo ER

  Se usa frecuentemente para el diseño conceptual de Bases de datos. El modelo ER se puede representar de la siguiente manera (TutorialsPoint, 2016):

Ilustración 11: Modelo Entidad Relación
Fuente: (TutorialsPoint, 2016)

- Descripción de tablas

  Cada tabla creada debe tener un nombre único en la Base de Datos, haciéndola accesible mediante su nombre o su seudónimo (Alias) (dependiendo del tipo de base de datos elegida).

  Las tablas son los objetos principales de bases de datos que se utilizan para guardar datos.

- Diccionario de datos

  El Diccionario de datos es la colección central de información sobre los datos. Almacena el significado y el origen de los datos, su relación con otros datos, su formato de uso etc. El diccionario de datos tiene rigurosas definiciones de todos los nombres para facilitar al Usuario y a los diseñadores de software (TutorialsPoint, 2016).

- Diccionario de clases

  En esta parte se describen todas las clases que componen el sistema especificando cada una con su nombre tipo y descripción

- Diccionario de atributos

  En esta parte se describen los atributos de las clases especificando su nombre, tipo, visibilidad y descripción.

- Diccionario de métodos de clase

  Aquí se describen las características de los métodos de cada una de las clases del sistema especificando nombre, tipo de retorno, parámetros y descripción.

2.3.3.3. Fase de Implementación y Pruebas Unitarias

IMPLEMENTACIÓN

La etapa de implementación de software se utiliza para producir un elemento del sistema especificado (elemento de software) e implementado en el software. Este proceso lo transforma en un comportamiento determinado, en interfaces y las limitaciones de aplicación en acciones de implementación que resulta en un elemento del sistema, que satisfaga los requerimientos del sistema (Palacios, 2015).

Cuando se implementa software, resulta recomendable comprobar que el código que hemos escrito funciona correctamente; es por ello que se busca realizar pruebas para corregir todos los errores antes de ser entregado al usuario final.

En las pruebas, el "probador" debe buscar situaciones límite que expongan las limitaciones de la implementación del componente, ya sea tratando éste como una caja negra ("pruebas de caja negra") o fijándonos en su estructura interna ("pruebas de caja blanca") (Berzal, 2017).

- Pruebas de caja negra***:***

  En este tipo de pruebas la estructura del programa es irrelevante o desconocida o, dicho de otra forma, es transparente para el probador ya que la importancia de este tipo de pruebas reside en la funcionalidad como tal. Es por ello, que en muchas ocasiones se las conoce como pruebas funcionales, porque valida el comportamiento de entrada y de salida de un cierto elemento.

  Quien desarrolla la prueba no tiene acceso o conocimiento de las líneas de código que se generaron para la funcionalidad del sistema, sólo sabe que para la entrada “x” se debe generar la salida “y“ (García I. M., 2017).

  En ellas se intenta encontrar errores de las siguientes categorías:

  - Funciones incorrectas o ausentes.
  - Errores de interfaz
  - Errores en estructuras de datos o en accesos a bases de datos externas.
  - Errores de rendimiento.
  - Errores de inicialización y de terminación.

Ilustración 12: Prueba de caja negra
Fuente: (García I. M., 2017).

- Pruebas de caja blanca: Estas pruebas también son conocidas como pruebas estructurales, éstas toman en cuenta el mecanismo interno del sistema o componente. Quien ejecuta la prueba sabe cómo fue

implementado el código y basado en esto diseña los casos de prueba (García I. M., 2017)

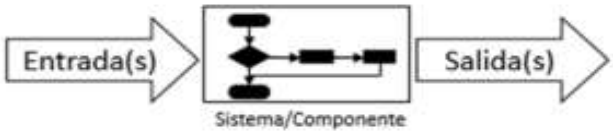


Ilustración 13: Prueba de caja blanca
Fuente: (García I. M., 2017)

PRUEBAS UNITARIAS

Las pruebas unitarias sirven para comprobar el correcto funcionamiento de un componente concreto de nuestro sistema (Berzal, 2017).

El software puede tener posteriores mantenimientos de código y, de ser necesario, parte de su código ser reutilizado para otras aplicaciones. (Durand, 2017).

Para realizar las pruebas unitarias debemos tener en cuenta o siguiente:

Tabla 17: Aspectos a tener en cuenta en una prueba unitaria

Test	Objetivo	Participantes	Ambiente	Método
Unitario	Detectar errores en los datos, lógica, algoritmos	Programadores	Desarrollo	Caja Blanca

Fuente: (Roldan, 2010)

Como muestra el cuadro es necesario tener en cuenta el ambiente en el cual se desarrollarán las pruebas es por ello que a continuación se darán a conocer algunas características que debe poseer el ambiente: (Morillo, 2012):

Características:

- Debe residir en un computador distinto al personal del desarrollador. Preferiblemente en un servidor compartido por los equipos de pruebas de software.
- Utilizar nombres de dominio (no direcciones IP) distintos a los de producción y desarrollo para evitar confusión del personal de pruebas.
- Ser lo más similarmente posible al ambiente de producción.

- Si no es posible tener instalado el mismo manejador de base de datos que en producción y desarrollo, automatizar la propagación de cambios de un ambiente a otro (Existen herramientas de software que lo permiten).

Para agilizar las pruebas resulta recomendable que un test sea completamente automático y compruebe los resultados esperados. No es muy apropiado llamar a una función, guardar el resultado en algún sitio y después tener que comprobar manualmente si el resultado era el deseado. En la práctica, mantener automatizado un conjunto amplio de tests permite reducir el tiempo que se tarda en depurar errores y en verificar la corrección del código (Galiano, 2017).

2.3.3.4. Fase integración y pruebas del sistema

INTEGRACIÓN

La integración hace referencia a una actividad de desarrollo de software que combina componentes de software diferentes en un conjunto. La integración se realiza en varios niveles y fases de la implementación.

- La integración del trabajo de un equipo que trabaja en el mismo subsistema de implementación antes de liberar el subsistema para los integradores del sistema.
- La integración de subsistemas en un sistema completo.

- Pruebas de Integración (Anexo 23 – Obs. 3)

Se ejecutan una vez concluidas las pruebas unitarias, teniendo todos o la mayor parte de los componentes integrados, para verificar que todos operen correctamente de manera conjunta. Se valida el software a través de varias interfaces y casos de uso tomando en cuenta que la salida de un componente es la entrada de otro.
En un sistema de tamaño medio a grande, estas pruebas regularmente son ejecutadas por los especialistas en pruebas(probador) con apoyo de los desarrolladores, requieren de una entidad que juegue el papel de mediador entre los diversos grupos que participan en estas pruebas.

A diferencia de las pruebas unitarias, las pruebas de integración, requieren una mejor estructura y organización, requiere al menos de un plan de pruebas (Testing, 2018).

Tabla 18: Aspectos a tener en cuenta en una prueba de integración

Test	Objetivo	Participantes	Ambiente	Método
Integración	Identificar errores introducidos por la combinación de programas probados unitariamente.	Programadores	Desarrollo	Caja Blanca y Negra

Fuente: Elaboración propia

La verificación de software es una disciplina de la ingeniería de software cuyo objetivo es asegurar que el software satisface por completo todos los requisitos esperados.

En la verificación el resultado final del desarrollo software debe concordar con los requerimientos del sistema, por lo que debemos asegurarnos que el desarrollo final coincida con dicha especificación.

PRUEBAS DEL SISTEMA

Las pruebas del sistema se ocupan del comportamiento de un sistema completo. La mayoría de los fallos funcionales deberían haber sido identificados antes, durante las fases de pruebas de unidad y pruebas de integración. Las pruebas de sistema se consideran normalmente como las apropiadas para comparar el sistema con los requisitos no funcionales del sistema, como seguridad, rendimiento, exactitud, velocidad y confiabilidad. (BIBIÁN, 2017)

Tabla 19: Aspectos a tener en cuenta en una prueba del sistema

Test	Objetivo	Participantes	Ambiente	Método
Sistema	Asegurar la apropiada navegación dentro del sistema, ingreso de datos, procesamiento y recuperación.	Programador	Desarrollo	Caja negra

Fuente: Elaboración propia

La prueba de Sistema incluye:

- Prueba funcionalidad: Consisten en la revisión de los requisitos aceptados por el cliente contra las funcionalidades presentes en la aplicación.
- Prueba de Usabilidad: Prueba enfocada a factores humanos, estéticos, consistencia en la interfaz de usuario, ayuda sensitiva al contexto y en línea, asistente documentación de usuarios y materiales de entrenamiento.
- Prueba de Confiabilidad Recuperación y tolerancia a fallas: Verificar que los procesos de recuperación (manual o automática) restauran apropiadamente la base de datos, aplicaciones y sistemas, y los llevan a un estado conocido o deseado.

Una vez que el sistema está completo, cumple con los requerimientos establecidos y con las pruebas de sistema aprobadas, en un nivel aceptable de calidad, debe estar disponible la documentación necesaria para ser llevado al entorno del usuario y produzca resultados positivos para el usuario y para el equipo de trabajo del proyecto, esto es a lo que se le conoce como la etapa de transición.

El proceso de transición a menudo se usa para implementaciones recurrentes de software en diferentes entornos, por ejemplo, de un entorno de desarrollo a un entorno de prueba o mantenimiento.

La transición puede darse de diferentes maneras; en los sistemas de software pueden implicar la reubicación física del hardware, la instalación (ésta necesita actividades de despliegue las cuales se muestran en el Anexo 21) y activación o desactivación de la infraestructura física o virtual. Otra manera de darse es sobre un cambio de un sistema antiguo a uno nuevo a menudo esto se realiza simultáneamente con la retirada y eliminación de un sistema existente, lo que implica la migración de datos. (Anexo 23 - Obs. 2)

Si se diera este último caso es necesario tener en cuenta unas pautas para la migración de datos:

- ✓ Tiempo que llevará realizar la migración completa.
- ✓ Cantidad de tiempo de inactividad que se requerirá.

- ✓ Riesgo para el negocio derivado de problemas técnicos de compatibilidad, corrupción de datos, problemas de rendimiento de aplicaciones y pérdida u omisión de datos.

Para minimizar el riesgo inherente al movimiento de datos, es preciso:

- ✓ Entender qué datos se está migrando, de qué tipo son, cuál es su origen y qué formato adquirirán en destino, una vez completado el traslado.
- ✓ Aplicar los procesos ETL (extracción, transformación y carga) preferiblemente antes de proceder a la migración.
- ✓ Definir e implementar políticas de migración de datos para garantizar el orden necesario a lo largo de todo el proceso.
- ✓ Apostar por las pruebas y validación de los datos migrados, por ser la única manera efectiva de asegurarse de que reúnen todos los atributos de calidad necesarios.

La validación en la ingeniería de software son el proceso de revisión que verifica que el sistema de software producido cumple con las especificaciones y que logra su cometido. Es normalmente una parte del proceso de pruebas de software de un proyecto, que también utiliza técnicas tales como evaluaciones, inspecciones y tutoriales.

La pregunta a realizarse es: ¿Es esto lo que el cliente quiere?

Características

- Comprobar que se satisfacen los requisitos.
- No hay programas de prueba, sino sólo el código final de la aplicación.
- Se prueba el programa completo.
- Uno o varios casos de prueba por cada requisito o caso de uso especificado.

Muchas veces la validación del software se confunde usualmente con la verificación de software. Las diferencias entre verificación y validación del software son:

- La verificación de software pregunta "¿Estamos construyendo correctamente el producto?"; esto es, ¿el software cumple con sus requisitos? (como una casa cumple con sus planos).

- La validación de software pregunta "¿Estamos construyendo el producto correcto?"; esto es, ¿hace el software lo que el usuario realmente requiere? (como una casa cumple con lo que el propietario necesita y quiere).

- *Pruebas de aceptación*

  El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.

  Estas pruebas van dirigidas a comprobar que el sistema cumple los requisitos de funcionamiento esperado, recogidos en el catálogo de requisitos y en los criterios de aceptación del sistema de información, y conseguir así la aceptación final del sistema por parte del usuario.

Tabla 20: Aspectos a tener en cuenta en una prueba de aceptación

Test	Objetivo	Participantes	Ambiente	Método
Aceptación	Detectar errores en los datos, lógica, algoritmos	Programadores	Desarrollo	Caja negra

Fuente: Elaboración propia

2.3.3.5. Fase de operación y mantenimiento

- **Operación**

  La fase de operación contempla las actividades y tareas del operador conforme se establece en el procedimiento de Pase a Producción, y cubre la operación del producto de software y el apoyo al usuario en esta operación.

  Se deberá capacitar al personal en la funcionalidad del sistema y entregar el documento de Manual de Usuario para la atención y apoyo a los usuarios en la operación de sistema.

- **Mantenimiento**

  La fase de mantenimiento de software involucra cambios al software en orden de corregir defectos y dependencias encontradas durante su uso

tanto como la adición de nueva funcionalidad para mejorar la usabilidad y aplicabilidad del software (SINCOWS, 2013).

A continuación, se señalan los tipos servicio de mantenimientos existentes:

Mantenimiento correctivo:

Tiene por objetivo localizar y eliminar los posibles defectos de los programas. Un defecto en un sistema es una característica del sistema con el potencial de provocar un fallo. Un fallo se produce cuando el comportamiento de un sistema difiere con respecto al comportamiento definido en la especificación.

Mantenimiento adaptativo:

Consiste en la modificación de un programa debido a cambios en el entorno (hardware o software) en el que se ejecuta. Desde cambios en el sistema operativo, pasando por cambios en la arquitectura física del sistema informático, hasta en el entorno de desarrollo del software. Este tipo de mantenimiento puede ser desde un pequeño retoque hasta una reescritura de todo el código.

Mantenimiento perfectivo:

Conjunto de actividades para mejorar o añadir nuevas funcionalidades requeridas por el usuario. Se divide en dos:

- Mantenimiento de Ampliación incorporación de nuevas funcionalidades.
- Mantenimiento de Eficiencia: mejora de la eficiencia de ejecución.

Mantenimiento preventivo:

Modificación del software para mejorar las propiedades de dicho software (calidad y mantenibilidad) sin alterar sus especificaciones funcionales.

El principal objetivo de este tipo de mantenimiento es mitigar o evitar las consecuencias de los fallos. Para ello (García S. M., 2012): (Anexo 23 – Obs. 5)

- ✓ Comprobación de la validez de los datos de entrada.

- ✓ Reestructuración del software para mejorar la legibilidad y su futuro mantenimiento.
- ✓ Adición de comentarios.
- ✓ Monitorización de las prestaciones del sistema en todo momento (Almacenamiento, Procesamiento, Ancho de Banda, etc.).

Otros Mantenimientos Vs. Preventivo:

- Los otros tipos de mantenimientos se realizan, generalmente a causa de una petición.
- El mantenimiento preventivo se produce tras un estudio de posibilidades de mejora en los diferentes módulos del sistema.

2.3.4. Norma ISO/IEC 25010

La evaluación de un software requiere tomar en cuenta algunos factores que son fundamentales en un proceso formal de selección, entre ellos: la funcionalidad del software; es decir, la forma como soporta los procesos y actividades con el uso de las TI (Morris, 2011).

Es por ello que se ha optó por utilizar la norma ISO/IEC 25010 que hace parte de la familia de normas ISO 25000. Es una norma que está centrada hacia la usabilidad, en el cual se determinan las características de calidad que se deben tener en cuenta en el momento de evaluar las propiedades de un producto software terminado.

Este modelo de calidad genérico clasifica a la calidad del producto, en características que se dividen en sub-características y atributos de calidad, el cual consiste de dos partes:

a) El modelo para la calidad interna y externa de un producto software.
b) El modelo para la calidad en uso de un producto software

Las métricas de calidad de software que serán definidas deben ser detalladas, con el objetivo de identificar las características de calidad del producto software más relevantes que se analizarán y se ejecutarán en la evaluación. Para ello, se utilizará una tabla de métricas, la cual contiene los siguientes ítems descritos en la Tabla 20. (Anexo 24 – Obs.2)

Tabla 21: Descripción de tabla de métricas

TABLA DE MÉTRICAS	
Ítem	**Descripción**
Sub Característica	Sub característica de calidad
Nombre de la métrica	Nombre asignado a la métrica de calidad.
Fase del Ciclo de vida del producto.	Fase del ciclo de vida: calidad interna, calidad externa y calidad en uso
Propósito de la métrica de calidad.	Motivo por el cual se selecciona la métrica.
Método de aplicación	Manera de cómo se va a aplicar la métrica
Formula y cálculo de datos	Establece la fórmula de medición y especifica los significados de los datos que se van a utilizar
Valor deseado	Proporciona el rango y los valores preferibles y recomendados.
Tipo de medida	Especifica en tipo de medida que se va seleccionar, como: tamaño (tamaño de la función, tamaño de la fuente), tiempo (lapso de tiempo, tiempo de usuario), contar (número de cambios, números de fallas).
Recursos utilizados	Específica los recursos que se utilizarán para poder medir cada métrica, entre los recursos utilizados, pueden estar: entrevistas a usuarios, código fuente, documentación, entre otras.

Fuente: ISO/IEC 25020

Para la presente tesis tomó en consideración el primer modelo porque define la totalidad de las características del producto software desde una perspectiva interna y externa la cual puede ser medida y evaluada como el grado en que satisface los requisitos de sus usuarios, aportando de esta forma valor.

El modelo define 8 características para la calidad interna y externa de un producto software: Adecuación Funcional, Fiabilidad, Eficiencia en el Desempeño, Facilidad de Uso, Seguridad, Compatibilidad, Mantenibilidad y Portabilidad, las cuales a su vez son subdivididos en sub-características descritas a continuación en la ilustración 16. Estas sub-características pueden ser medidas con métricas internas o externas que se muestran en el Anexo 2.

Ilustración 14: Modelo de calidad del producto de Software
Fuente: (Rodríguez, 2015)

2.3.3.1. Métricas de Calidad del Producto Software (Calidad Interna y Externa)

Las métricas para la calidad interna y externa evalúan las características que se definieron en la sección del Modelo de Calidad del Producto Software (Calidad Interna y Externa); A continuación, se muestra el siguiente cuadro el cual contiene únicamente las características que serán medidas en la presente tesis para la demostración de la hipótesis.

Tabla 22: Características elegidas del Modelo de calidad para Calidad Externa e Interna

CARACTERÍSTICAS	SUB CARACTERÍSTICAS	DESCRIPCIÓN
Funcionalidad	Pertinencia	Capacidad del producto de software para proveer un conjunto apropiado de funciones para tareas y objetivos de usuario específicos.
	Corrección	Capacidad del producto de software para proveer los resultados o efectos correctos o acordados, con el grado necesario de precisión.
	Completitud	Capacidad del producto de software para adherirse a estándares, normas y buenas prácticas relacionadas con funcionalidad.
Rendimiento	Comportamiento temporal	Capacidad de un sistema software para proporcionar los tiempos de respuesta y procesamiento apropiados.
	Utilización de recursos	Capacidad en que un sistema software utiliza las cantidades y tipos de recursos adecuados.
	Capacidad	Capacidad de un sistema software de cumplir con los requisitos determinados.
Fiabilidad	Madurez	Capacidad del sistema para satisfacer las necesidades de fiabilidad durante el funcionamiento normal.
	Disponibilidad	Capacidad de un sistema de estar operativo y accesible para su uso cuando se necesite.
	Tolerancia a fallos	Capacidad de un sistema para operar cuando se presenten fallos.
	Capacidad de recuperación	Capacidad de un sistema para restablecer el estado del sistema y recuperar datos que se hayan afectado, en caso de fallo.
Usabilidad	Inteligibilidad	Capacidad del sistema software que permite al usuario entender si el software es adecuado para sus necesidades
	Aprendizaje	Capacidad del sistema, que permite al usuario entender si el software es adecuado para alcanzar sus objetivos determinados
	Operabilidad	Capacidad de un sistema software que permite al usuario operarlo y controlarlo con facilidad.
	Protección a errores	Capacidad en que el sistema brinda la protección necesaria contra errores que realizan los usuarios
	Atractividad	Capacidad en que la interfaz de usuario llega a satisfacer y agradar al usuario.
	Accesibilidad	Capacidad del sistema software para que se permita ser utilizado por usuarios específicos.

CARACTERÍSTICAS	SUB CARACTERÍSTICAS	DESCRIPCIÓN
Mantenibilidad	Modularidad	Capacidad del producto de software que permite una determinada modificación sea implementada sin afectar otras funcionalidades del producto de software.
	Analizabilidad	Facilidad con la que se puede llevar a cabo un análisis del impacto de una determinada modificación en el sistema.
	Capacidad de ser reusado	Capacidad de un activo (Información, Software, Hardware, Usuarios) para ser utilizado en más de un sistema o en la construcción de otros activos.
	Capacidad de ser modificado	Capacidad del sistema para permitir que sea modificado sin causar daños o reducir la calidad del producto existente.
	Capacidad de ser probado	Facilidad de realizar pruebas a un sistema o componente software, para determinar si se han cumplido con los requerimientos establecidos.

Fuente: (Carazas, 2015)

Ilustración 15: Modelo de calidad para Calidad Externa e Interna

Fuente: Propia

Mapeo de las Etapas de la Metodología en Cascada con los Procesos de la Norma ISO 12207:2017

Tabla 23: Etapas de la Metodología en Cascada con los Procesos Norma ISO 12207:2017

PROCESOS ISO 12207:2017		**PROCESOS DEL MODELO CASCADA**
Cláusula	**Sub Cláusulas**	**Nombre del proceso**
6.1 Procesos de acuerdo	6.1.1 Proceso de adquisición	No se incluyeron estos procesos
	6.1.2 Proceso de suministro	
6.2 Procesos de habilitación de proyectos organizacionales	6.2.1 Proceso de gestión del modelo de ciclo de vida.	
	6.2.2 Proceso de gestión de la infraestructura.	
	6.2.3 Proceso de gestión de cartera.	
	6.2.4 Proceso de gestión de recursos humanos.	
	6.2.5 Proceso de gestión de la calidad.	
	6.2.6 Proceso de gestión del conocimiento	
6.3 Procesos de gestión técnica.	6.3.1 Proceso de planificación del proyecto	
	6.3.2 Proceso de evaluación y control del proyecto.	
	6.3.3 Proceso de gestión de decisiones	
	6.3.4 Proceso de gestión de riesgos.	
	6.3.5 Proceso de gestión de la configuración.	
	6.3.6 Proceso de gestión de la información.	
	6.3.7 Proceso de medición	
	6.3.8 Proceso de aseguramiento de la calidad.	

PROCESOS ISO 12207:2017		PROCESOS DEL MODELO CASCADA
Cláusula	**Sub Cláusulas**	**Nombre del proceso**
6.4 Procesos técnicos	6.4.1 Análisis de negocio o misión	No se incluyó este proceso
	6.4.2 proceso de definición Necesidades y requisitos de los interesados	Análisis y Definición de Requerimientos
	6.4.3 proceso Definición de requisitos del sistema / software	
	6.4.4 Proceso de definición de arquitectura	Fase Diseño del Sistema
	6.4.5 Proceso de definición del diseño	
	6.4.6 Análisis del sistema	
	6.4.7 Proceso de implementación	Fase de implementación y pruebas de unidades
	6.4.8 Proceso de integración	Fase Integración y pruebas del sistema
	6.4.9 Proceso de verificación	
	6.4.10 Proceso de transición	Fase de transición y validación
	6.4.11 Proceso de validación	
	6.4.12 Proceso de operación	Fase de Operación y mantenimiento
	6.4.13 Proceso de mantenimiento	
	6.4.14 Proceso de eliminación	No se incluyó este proceso

Fuente: Elaboración Propia.

En esta parte podemos observar un cuadro relacional entre los procesos de la ISO 12207:2017 y el modelo de cascada cabe resaltar que este modelo ya se ha descrito anteriormente y es el que más se adecua a la forma de trabajo del área de Desarrollo de software de la MPCH. Por consiguiente como se puede observar el modelo de cascada se relaciona con los procesos técnicos de la ISO 12207:2017 es por ello que se ha decidido trabajar este proyecto de tesis tomando únicamente dichos procesos, por otro lado, los procesos excluyentes se encuentran enfocados a empresas o entidades que su negocio es netamente el desarrollo de software para otras empresas y debido a esto necesitan un estudio más interno de la viabilidad del proyecto para la creación del software, situación contraria con los procesos incluyentes ya que solo se deben a que la institución únicamente desarrolla sistemas para áreas internas

y da mantenimiento o realiza cambios a los sistemas ya existentes en caso lo requieran; y además son exigidos por normativas nuevas o impuestas ya que la MPCH es una Institución pública.

El área de desarrollo de sistemas de la MPCH debe de estar preparada en todos los sentidos para llevar a cabo exitosamente un proyecto de desarrollo de software interno, el cual representa en la mayoría de los casos una importante inversión y debe de minimizar los riesgos al máximo a fin de generar una ventaja competitiva al satisfacer las necesidades del negocio, a la par que mantiene, crece y actualiza los sistemas.

Desde la definición del alcance del proyecto, el área responsable debe de contar con experiencia en procesos de definición de perfiles, estimación, diseño, planeación, desarrollo, seguimiento, pruebas, etc.; los cuales deberán de trabajar en conjunto con el área que requiere el desarrollo.

2.4. Definición de la terminología

a) Ciclo de Vida del software

El ciclo de vida del desarrollo Software, es una secuencia estructurada y bien definida de las etapas en Ingeniería de software para desarrollar un producto, se podría decir que es un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando la vida del sistema desde la definición hasta la finalización de su uso.

b) Gestión de proyecto

Gestión de proyectos es un enfoque metódico para la planificación, organización, colaboración, evaluación y control en el desarrollo de un sistema aceptable con un costo mínimo y dentro de un período de tiempo específico.

c) Metodología de desarrollo de software

Son marcos de trabajo utilizados para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información.

d) Modelo de proceso de software

Es una representación simplificada de un proceso de software o también llamadas abstracciones útiles para explicar diferentes enfoques del desarrollo de software. Es una plantilla, patrón o marco que define el proceso a través del cual se crea software.

e) Ingeniería de software

Conjunto estructurado de programas que garantizan el desarrollo de una línea de especialización en el uso de métodos y técnicas que permitan desarrollar y mantener un software de calidad, diseñado para transmitir los conocimientos de forma organizada, gradual y consistente al perfil, con el objetivo de utilizar metodologías que garanticen el producto final.

f) Producto de Software

Es el conjunto de programas (fuentes y ejecutables), procedimientos, reglas y documentación posible asociada, así como los datos pertenecientes a la operación del sistema.

g) Calidad del Producto de software

Es el grado en el que un cliente percibe que el software cumple con sus expectativas.

h) Línea Base

Especificación o producto que ha sido formalmente revisado y acordado, que posteriormente sirve como base para un mayor desarrollo, y que solo se puede modificar a través de procedimientos formales de control de cambio.

i) Proceso

Conjunto de actividades interrelacionadas o que interactúan, que transforma los insumos en productos.

j) Stakeholder

Persona interesadas o involucradas en el desarrollo de un sistema, bajo una perspectiva. Esta puede ser económica o relacionada otro beneficio por el desarrollo del sistema.

k) Componente del software

Es una unidad modular (subsistemas o módulos) de un programa software con interfaces y dependencias bien definidas que permiten ofertar o solicitar un conjunto de servicios o funcionales.

l) Unidad del Software

Parte del software que realiza algún cometido, puede ser una función, un método, una clase, una biblioteca (librería), etc.

m) Elemento del Software

Se refiere a la parte inmaterial del software (archivos, registros, campos, datos, etc.)

n) Plan de Pruebas

Es un producto formal que define los objetivos de la prueba de un sistema, establece y coordina una estrategia de trabajo, y provee del marco adecuado para elaborar una planificación paso a paso de las actividades de prueba.

o) Validación

Demuestra que el producto, componentes del producto y artefactos corresponden a lo esperado para su uso.

p) Verificación

Demuestra que el producto, componentes del producto y artefactos cumplen con los requerimientos establecidos.

q) Atributo:

Propiedad inherente de una entidad que puede distinguirse cuantitativa o cualitativamente.

r) Calidad interna:

Capacidad de un conjunto estático de atributos para satisfacer las necesidades declaradas e implícitas de un producto software bajo ciertas condiciones especificadas.

s) Calidad externa:

Capacidad de un producto software para desarrollar el comportamiento de un sistema de forma que satisfaga las necesidades declaradas e implícitas de un sistema utilizado bajo ciertas condiciones especificadas.

t) Métrica:

Medida, tanto base como derivada, utilizada para medir la calidad del software.

u) Técnica:

Procedimiento o conjunto de reglas, normas o protocolos que tienen como objetivo obtener un resultado determinado.

v) Herramienta:

Instrumento más o menos simple destinado a realizar un cierto trabajo.

w) Método:

Es un conjunto de herramientas, técnicas y procesos que brindan soporte y facilitan el logro u obtención de una meta

CAPÍTULO III: MARCO METODOLÓGICO

3.1. Hipótesis

Debido a que la investigación es descriptiva propositiva, no se platea hipótesis. En su reemplazo se platean las siguientes preguntas de investigación:

a. ¿De qué forma el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, se adecúa funcionalmente a las tareas y objetivos del proceso de desarrollo de software en la Municipalidad?
b. ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, permite niveles de usabilidad aceptables, tanto por los usuarios de TI como del personal de desarrollo del área de TI, en la Municipalidad?
c. ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 permite niveles aceptables de mantenibilidad de las aplicaciones que están en producción en la Municipalidad?
d. ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 brinda fiabilidad a los usuarios en el funcionamiento diario y operatividad cuando ocurran fallos en las aplicaciones que están en producción en la Municipalidad?
e. ¿De qué manera el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017 permite niveles aceptables de rendimiento para proporcionar adecuados tiempos de respuesta y procesamiento en las aplicaciones que están en producción en la Municipalidad?

3.2. Tipo de Investigación

El tipo de investigación se clasifica en:

a. Según su alcance o el nivel de conocimientos que se adquieren:

- **Descriptiva,** porque en esta investigación únicamente llegará a un nivel de diseño de nuestra metodología propuesta es decir se detallará toda la procedimentación de cómo ejecutar correctamente el proceso de desarrollo de software en la MPCH.

- **Propositiva,** porque esta investigación no pretende implementar el método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, por las limitaciones que se tienen. Por tanto, quedará como una propuesta de solución a los problemas en el proceso de desarrollo de software en la empresa MPCH.

b. Según el propósito o finalidad perseguida

- **Aplicada**, debido a que se pretende ampliar conocimientos científicos aplicando el marco de referencia NTP ISO 12207:2017, buscando así mejorar el proceso de desarrollo de software en la MPCH.

c. Según los medios utilizados para obtener los datos:

- **No Experimental,** porque las variables de estudio no se manipularán deliberadamente y no se pretende modificar la situación actual del proceso de desarrollo de software en la MPCH.

3.3. Estrategia para la demostración de la hipótesis

3.3.1. Variables e indicadores

Al ser la investigación descriptiva, no se evaluará el efecto de una variable independiente sobre la variable dependiente. Por tanto, no existe variable dependiente.

Sólo se validará el modelo para gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017, el cual será evaluada por un juicio de expertos tomando como referencia para la evaluación los criterios de la ISO/IEC 25010, en los siguientes factores: funcionalidad, usabilidad, rendimiento, fiabilidad y mantenibilidad.

Los expertos que fueron considerados para dicha evaluación fueron los siguientes:

Tabla 24 Identificación de expertos para la valoración del método para la gestión del desarrollo del software propuesto, basada en la NTP ISO 12207:2017

	Experto 1	Experto 2	Experto 3
Nombres y Apellidos	Erwin Mac Dowall Reynoso	Junior Eugenio Cachay Maco	Oscar Zocón Alva
Formación académica	- Licenciado en Computación - Maestro en Computación	- Ingeniero en computación e informática - Maestría en Ingeniería de Sistemas con Mención en Gerencia de Tecnologías de Información y Gestión del Software	- Ingeniero de Computación y Sistemas - Magister en Ingeniería de Sistemas
Área de experiencia profesional	- Especialización en Sistemas de Gestión de la Seguridad de la Información ISO 27001 - Certificación ITIL Fundation	- Cursos y seminarios en Formación de Auditores Internos bajo la Norma ISO 9001:2008 y - Introducción y Sensibilización en Gestión de Calidad ISO 9001:2008	- Certificación Internacional ITIL en Gestión de Servicios de Tecnologías de Información - Certificación en Gestión y Desarrollo de Proyectos Agiles Certified Scrum Developer - Auditor Interno ISO 27001:2007
Tiempo de experiencia	31 años	11 años	23 años
Cargo actual	- Consultor de TI - Docente Universitario	- Gerente general	- Docente Universitario
Institución	- Independiente	- Audit and control of information systems sac	- Universidad Nacional de Cajamarca

Los objetivos perseguidos con la validación del método propuesto para la gestión de Desarrollo de Software fueron:

a. Objetivo de la investigación

Desarrollar un método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 que permita mejorar el proceso de desarrollo de software en la Municipalidad Provincial de Chiclayo.

b. Objetivo del juicio de expertos

Validar el método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 en relación a la funcionalidad, fiabilidad, usabilidad, rendimiento y mantenibilidad.

Los criterios y el sistema de valoración del método propuesto para la gestión de desarrollo de software, basada en la NTP ISO 12207:2017, fueron:

Tabla 25 Criterios y sistema de valoración del método propuesto para la Gestión del Desarrollo del Software, basada en la NTP ISO/IEC 12207:2017 en la Municipalidad Provincial de Chiclayo, por juicio de expertos

CATEGORIA	CALIFICACIÓN	INDICADOR
FUNCIONABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de proveer los resultados, efectos correctos o acordados, con el grado necesario de precisión.	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes obtener los resultados esperados en el producto de software
	2. Bajo Nivel	La actividad o tarea permiten obtener algunos efectos correctos, pero no corresponden con la totalidad de obtener los resultados esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.
	4. Alto nivel	La actividad o tarea de la metodología permiten la correcta funcionalidad del producto de software.
CATEGORIA	**CALIFICACIÓN**	**INDICADOR**
USABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir al usuario aprender el manejo del producto de software operarlo y controlarlo.	1. No cumple con el criterio	La actividad o tarea de la metodología no permite la correcta usabilidad en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para lograr su correcto uso
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr que el usuario maneje opere y controle el producto de software.
	4. Alto nivel	La actividad o tarea permite la correcta usabilidad del producto de software
MANTENIBILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de evitar efectos inesperados a causa de modificar el producto de software	1. No cumple con el criterio	La actividad o tarea no es capaz de soportar modificaciones en el producto de software
	2. Bajo Nivel	La actividad o tarea es poco probable que soporte modificaciones en el producto de software
	3. Moderado nivel	La actividad o tarea acepta efectos inesperados en el producto de software
	4. Alto nivel	La actividad o tarea se encuentra completamente apta para adaptarse a modificaciones ante algún efecto inesperado.
RENDIMIENTO Los aspectos considerados en la actividad o tarea de la	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes cumplir con los requisitos determinados.

metodología son capaces de proporcionar los tiempos de respuesta y procesamiento apropiados, además de cumplir con los requisitos determinados.	2. Bajo Nivel	La actividad o tarea permiten obtener algunos tiempos de respuesta adecuados, pero no corresponden con la totalidad de obtener los requisitos esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.
	4. Alto nivel	La actividad o tarea de la metodología permiten proporcionar tiempos de respuesta y procesamiento adecuados, y cumplir con los requisitos.
FIABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir la operabilidad aun cuando se presenten fallos	1. No cumple con el criterio	La actividad o tarea de la metodología no da fiabilidad a los usuarios en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para que opere normalmente el producto de software.
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr la fiabilidad en la operabilidad del producto de software aun cuando presente fallos.
	4. Alto nivel	La actividad o tarea permite que el producto de software sea fiable para los usuarios al momento de usar el producto de software.

El modelo conceptual de la investigación que se grafica a continuación, muestra las variables de la investigación y las dimensiones que se evaluarán para contrastar la hipótesis:

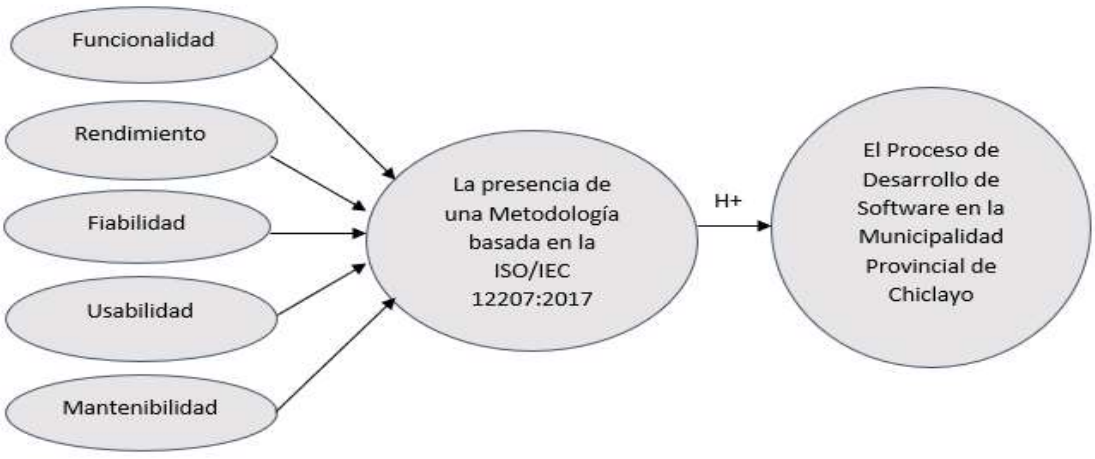


Ilustración 16: Modelo conceptual de la Investigación
Fuente: Elaboración propia

Tabla 26: *Operacionalización de variables*

VARIABLE	DIMENSION	TITULO DEL INDICADOR	ESCALA
INDEPENDIENTE Metodología de Desarrollo de Software basada en la NTP ISO 12207:2017	Funcionabilidad	Pertinencia	Likert de 4 niveles
		Corrección	Likert de 4 niveles
		Completitud	Likert de 4 niveles
	Rendimiento	Comportamiento temporal	Likert de 4 niveles
		Utilización de recursos	Likert de 4 niveles
		Capacidad	Likert de 4 niveles
	Fiabilidad	Madurez	Likert de 4 niveles
		Disponibilidad	Likert de 4 niveles
		Tolerancia a fallos	Likert de 4 niveles
		Capacidad de recuperación	Likert de 4 niveles
	Usabilidad	Inteligibilidad	Likert de 4 niveles
		Aprendizaje	Likert de 4 niveles
		Operabilidad	Likert de 4 niveles
		Protección a errores	Likert de 4 niveles
		Atractividad	Likert de 4 niveles
		Accesibilidad	Likert de 4 niveles
	Mantenibilidad	Modularidad	Likert de 4 niveles
		Analizabilidad	Likert de 4 niveles
		Capacidad de ser reusado	Likert de 4 niveles
		Capacidad de ser modificado	Likert de 4 niveles
		Capacidad de ser probado	Likert de 4 niveles

3.4. Técnicas e instrumentos de recolección de datos

Las técnicas y sus respectivos instrumentos que usaremos para nuestra recolección de datos son las siguientes:

Tabla 27: Tabla de las técnicas e instrumentos a utilizar

TÉCNICA	INSTRUMENTO
Juicio de expertos Es un conjunto de opiniones que pueden brindar profesionales expertos en una industria o disciplina, relacionadas al proyecto que se está ejecutando.	**Cuestionario** Fue Dirigido a 3 profesionales expertos de Desarrollo de Software, para evaluar cualitativamente la propuesta y de esta forma dar a conocer que se cumplió con el objetivo principal para el que fue desarrollado.

TÉCNICA	INSTRUMENTO
Entrevista Es una técnica de recolección de información mediante una interrogación estructurada o una conversación totalmente libre; en ambos casos se utiliza un formulario o esquema con preguntas o cuestiones para enfocar la charla que sirven como guía.	**Formulario** Fue dirigido a 2 profesionales en el Desarrollo de Software de la MPCH, para tener una visión más clara de cómo llevaron a lo largo de su experiencia el proceso de desarrollo de software y así complementar nuestra investigación.
Análisis de Información Es una forma de investigación, cuyo objetivo es la captación, evaluación, selección y síntesis de los mensajes en el contenido de los documentos buscando obtener ideas relevantes, de las distintas fuentes de información.	**Documentos web, Tesis, Libros, artículo periodístico, normativa de la MPCH** Son las fuentes utilizadas como base para elaborar la metodología de Desarrollo de Software propuesta.

Fuente: Elaboración propia

CAPÍTULO IV: DESARROLLO DE LA PROPUESTA

4.1. Modelo de aplicación basado en la Norma ISO/IEC 12207, al Proceso de Desarrollo de Software en la Municipalidad de Chiclayo

4.1.1. Fase de análisis y definición de requerimientos

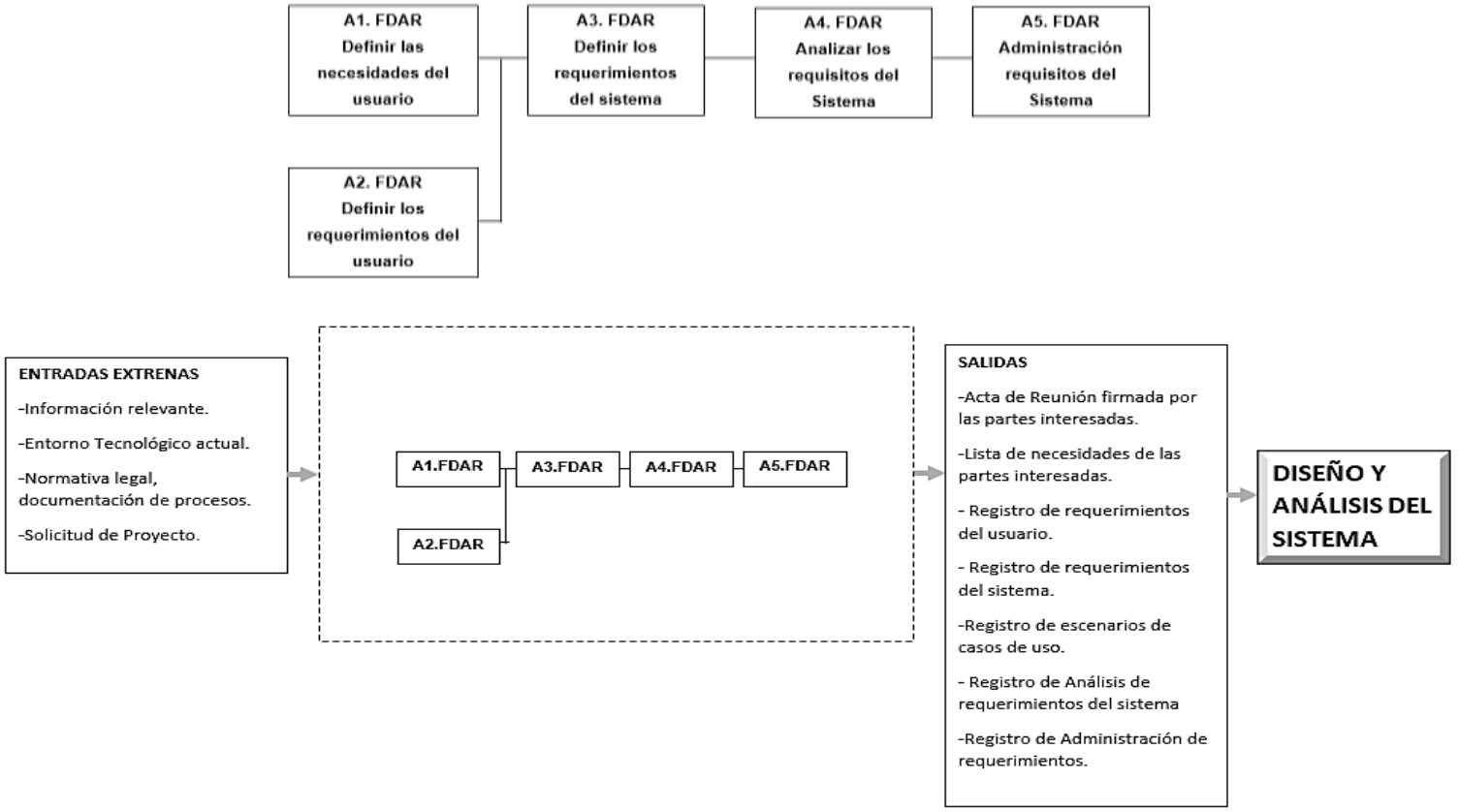


Ilustración 17*:* Fase de análisis y definición de requerimientos

Fuente: Elaboración propia

<table>
<tr><td>Proceso 6.4.2
ISO 12207:2017</td><td>Definición de requisitos y necesidades de las partes interesadas</td><td>Código de la Actividad:</td><td>A1P6.4.2</td><td>Actividad:</td><td>Definir las necesidades del usuario</td></tr>
<tr><td>Objetivo de la actividad:</td><td colspan="5">Identificar las necesidades de las partes interesadas y evaluar su nivel de importancia, ya que estas influyen en la efectividad en el desarrollo del software.</td></tr>
<tr><td>Entradas:</td><td colspan="5">Solicitud de requerimientos propia del área solicitante.
Documento de Funciones de la unidad solicitante, normativa legal, documentación de procesos.</td></tr>
<tr><td>N°</td><td colspan="2">Tareas</td><td>Responsable</td><td colspan="2">Artefactos</td></tr>
<tr><td>1</td><td colspan="2">Realizar un estudio sobre el sistema solicitado y si éste puede desarrollarse dentro de las restricciones presupuestales existentes</td><td>Analista de sistemas</td><td colspan="2" rowspan="5">- Formato de Acta de Reunión (Anexo 8)
- Formato de solicitud de Requerimiento (Anexo 4)
- Formato de Entrevista (Anexo 5)
- Formato de identificación de necesidades (Anexo 6)</td></tr>
<tr><td>2</td><td colspan="2">Convocar a una reunión con el área solicitante</td><td>Jefe de sistemas</td></tr>
<tr><td>3</td><td colspan="2">Identificar las necesidades de las partes interesadas.</td><td rowspan="3">Desarrollador</td></tr>
<tr><td>4</td><td colspan="2">Priorizar las necesidades las partes interesadas.</td></tr>
<tr><td>5</td><td colspan="2">Definir las necesidades y justificación de las partes interesadas.</td></tr>
<tr><td>Salidas</td><td colspan="5">Acta de Reunión firmada por las partes interesadas.
Entrevista de los usuarios
Lista de necesidades de las partes interesadas.</td></tr>
</table>

Proceso 6.4.2 ISO 12207:2017	Definición de requisitos de las partes interesadas	Código de la Actividad:	A2P6.4.2	Actividad:	Definir los requerimientos del usuario
Objetivo de la actividad:	Definir los requisitos de usuario como declaraciones formales a partir de las necesidades definidas por los usuarios.				
Entradas:	Acta de Reunión firmada por las partes interesadas. Entrevista de los usuarios Lista de necesidades de las partes interesadas.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Identificar las restricciones del sistema			Analista de Sistemas	Formato de Definición y análisis de Requerimiento de usuario. (Anexo 7)
2	Definir Deficiencias o debilidades encontradas.				
3	Validar que sus necesidades y expectativas del usuario se han capturado y expresado adecuadamente.				Acta de Conformidad de usuario (Anexo 12)
4	Analizar qué objetivo estratégico apoyan dichos requisitos (Anexo 25 – Obs.1)				
Salidas	Registro y análisis de requerimientos del usuario Acta de conformidad del usuario				

Proceso 6.4.3 ISO 12207:2017	Definición de requisitos del Sistema	Código de la Actividad:	A1P6.4.3	Actividad:	Definir los requerimientos del sistema
Objetivo de la actividad:	Definir claramente las capacidades deseadas en una vista técnica de una solución que satisfaga las necesidades operativas de usuario para llevar a cabo de manera adecuada las siguientes etapas del desarrollo del software.				
Entradas:	Acuerdo de conformidad de usuario Registro y análisis de requerimientos del usuario				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Definir los requerimientos del sistema (funcionales, no funcionales (atributos de calidad, interfaces externas, restricciones))			Analista de Sistemas	- Formato detallado Requerimiento del Sistema (Anexo 9) - Formato de especificación de requerimientos (Anexo 11)
2	Describir los requerimientos funcionales en casos de uso, historias de usuarios o escenarios, para lograr cumplir con los usuarios.				
3	Definir las restricciones de implementación necesarias.				
Salidas	Registro de requerimientos del sistema Registro de especificación de requerimientos				

Proceso 6.4.3 ISO 12207:2017	Definición de requisitos del Sistema	Código de la Actividad:	A2P6.4.3	Actividad:	Analizar los requisitos del Sistema
Objetivo de la actividad:	Analizar que los requerimientos cumplan con características de factibilidad, asequibilidad, equilibrio y otros, evitando así inconsistencias posteriormente.				
Entradas:	Registro de requerimientos del sistema Registro de especificación de requerimientos				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Analizar el conjunto completo de requerimientos del sistema.			Analista de Sistemas	Formato de definición y Análisis de Requerimientos del Sistema. (Anexo 10)
2	Evaluar la factibilidad de cada requerimiento				
3	Priorizar los requerimientos del Sistema				
4	Identificar las restricciones en una solución de sistema				
5	Identificar los problemas, deficiencias, conflictos y debilidades dentro del conjunto completo de requerimientos.				
6	Gestionar la Aceptación de los requerimientos				
Salidas	Registro de Análisis de requerimientos del sistema				

Proceso 6.4.3 ISO 12207:2017	Definición de requisitos del Sistema	Código de la Actividad:	A3P6.4.3	Actividad:	Administración de requerimientos del sistema
Objetivo de la actividad:	Definir, registrar y controlar la línea de base, generalmente bajo administración de configuración formal, junto con la administración de cambios resultantes de la aplicación de otros procesos del ciclo de vida como la arquitectura o el diseño.				
Entradas:	Registro de Análisis de requerimientos del sistema				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Realizar el control de versiones ante cualquier cambio después que un conjunto de requerimientos ha sido aprobado.			Analista de Sistemas	Formato de petición de cambio (Anexo 13)
2	Realizar el control de cambios si la petición de cambio ha sido aprobada; antes de ser implementada.				
3	Registrar la trazabilidad y dar seguimiento a cada requerimiento y garantizar que todas las necesidades del usuario sean atendidas.				Formato de especificación requerimientos (Anexo 11)
4	Asegurar que los requerimientos han sido implementados de forma completa.				
Salidas	Registro de petición y control de cambios Registro de especificación de los requerimientos del sistema.				

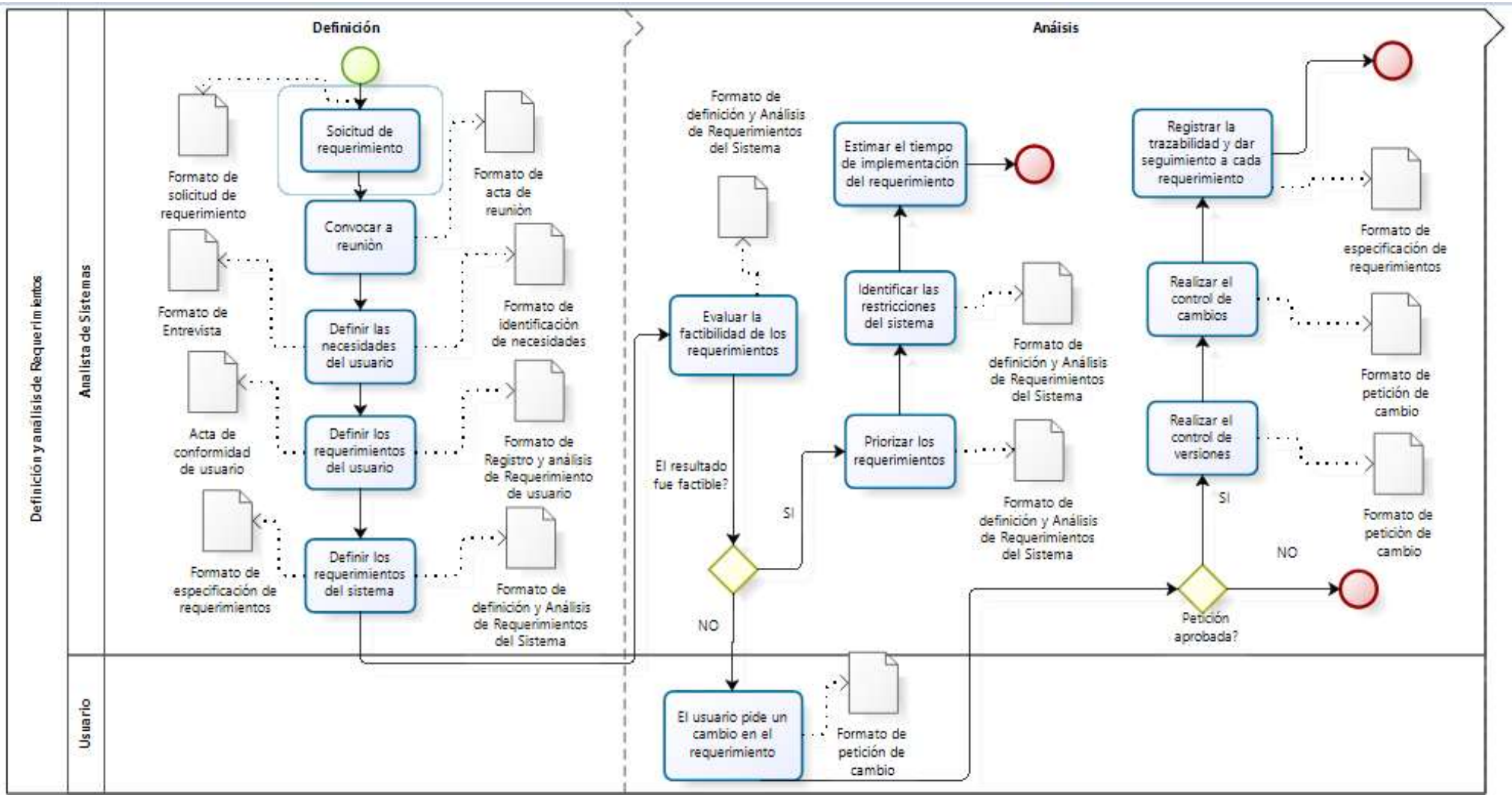


Ilustración 18: Diagrama – Análisis y Definición de requerimientos

Fuente: Elaboración propia

4.1.2. Fase análisis y diseño del sistema

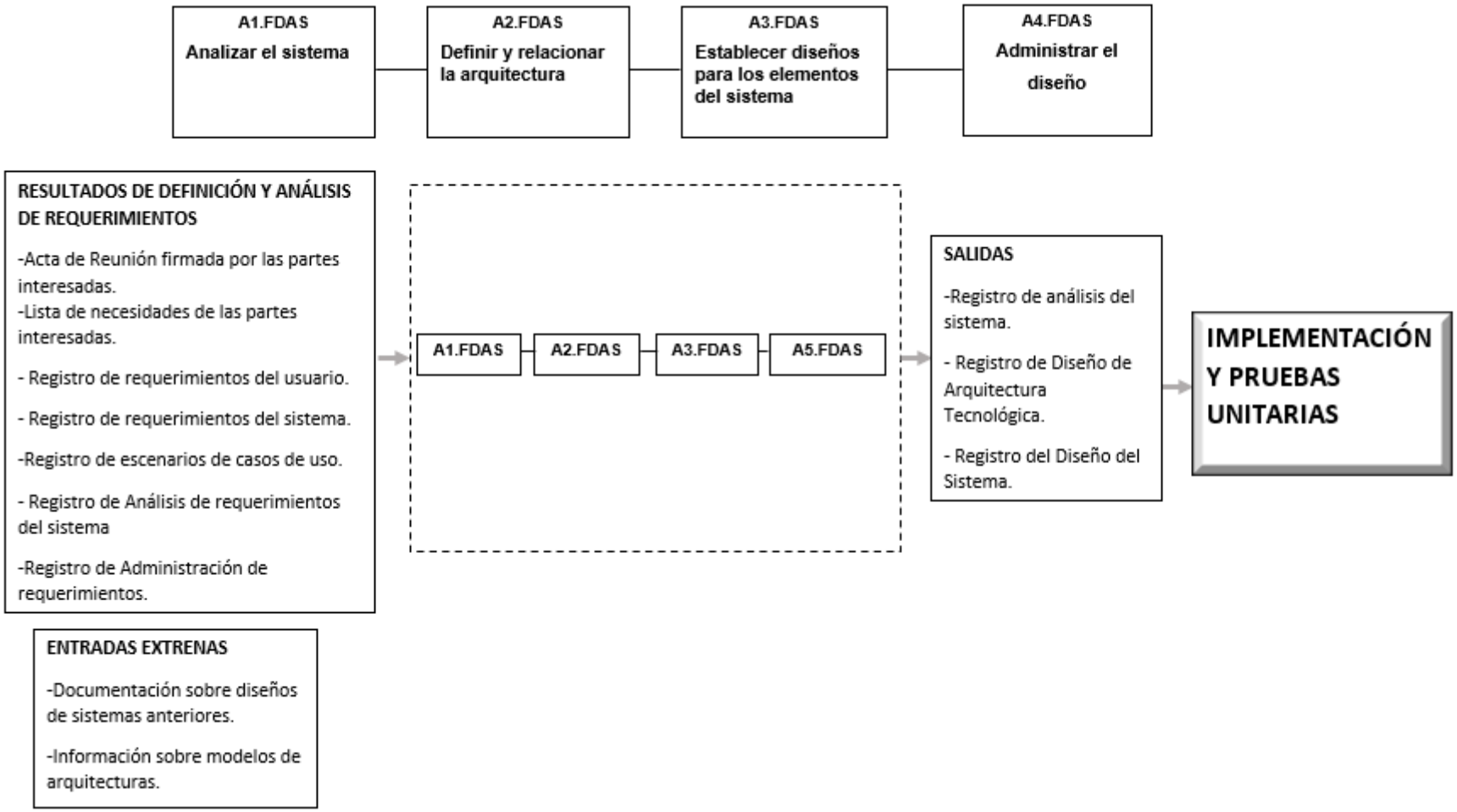


Ilustración 19*:* Fase del diseño y análisis del sistema
Fuente: Elaboración propia

Proceso 6.4.6 ISO 12207:2017	Proceso de análisis del sistema	Código de la Actividad:	A1P6.4.6	Actividad:	Analizar el Sistema
Objetivo de la actividad:	Realizar un seguimiento de la información del sistema para la comprensión técnica para ayudar a la toma de decisiones a lo largo del ciclo de vida.				
Entradas:	Registro de requerimientos de usuario Diagramas de contexto (representación general del proceso del negocio) Diagramas para el desarrollo estructurado (diagramas de flujo, diagrama entidad-relación). Registro de definición y Análisis requerimientos del sistema Registro de petición y control de cambios Registro de especificación de los requerimientos del sistema.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Definir los objetivos no funcionales como las características críticas de calidad.			Analista de Sistemas	Formato de análisis del Sistema (Anexo 16)
2	Revisar los documentos de salidas de la primera fase				
3	Analizar el objetivo estratégico de cada requerimiento y plantear opciones de arquitectura y diseño				
4	Registrar los resultados del análisis del sistema.				
5	Establecer conclusiones y recomendaciones.				
Salidas	Registro de análisis del Sistema				

Proceso 6.4.4 ISO 12207:2017	Definición de arquitectura	Código de la Actividad:	A1P6.4.4	Actividad:	Definir y relacionar la arquitectura
Objetivo de la actividad:	Representar la arquitectura que cumpla con los requisitos del sistema, y utilizar los elementos del sistema como medio para transmitir la intención arquitectónica y para verificar la viabilidad del diseño.				
Entradas:	Registro de especificación de los requerimientos del sistema. Registro y análisis del sistema				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Representar la arquitectura del sistema de software utilizando el enfoque del modelo 4+1 vistas que usa un lenguaje de descripción de arquitectura UML.			Analista de Sistemas	Formato de Diseño de Arquitectura Tecnológica (Anexo 14)
2	Identificar los elementos del sistema (hardware, software, comunicaciones) que se relacionan con las interfaces.				
3	Describir las restricciones técnicas derivadas de la tecnología seleccionada que afectan al diseño				
Salidas	Registro de Diseño de Arquitectura Tecnológica				

Proceso 6.4.5 ISO 12207:2017	Proceso de diseño	Código de la Actividad:	A1P6.4.5	Actividad:	Establecer diseños para los elementos del sistema.
Objetivo de la actividad:	Proporcionar suficientes datos detallados e información sobre el sistema y sus elementos para permitir posteriormente la correcta implementación.				
Entradas:	Información sobre alternativas de diseño Registro de Diseño de Arquitectura Tecnológica				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Revisar documentos sobre arquitectura de software, es decir los diferentes patrones que existen, y sobre las arquitecturas utilizadas en sistemas anteriores.			Analista de Sistemas	Formato de diseño del sistema (Anexo 15)
2	Decidir qué patrón de arquitectura se va a utilizar de acuerdo con las características que requiera el nuevo sistema.				
3	Definir las interfaces, los componentes y base de datos del sistema.				
4	Establecer los artefactos de diseño. (prototipos, modelos de datos, pseudocódigo, diagramas de relación de entidad, casos de uso, especificaciones de interfaz, diagramas de vistas de arquitectura)				
Salidas	Registro del Diseño del Sistema				

Proceso 6.4.5 ISO 12207:2017	Proceso de diseño	Código de la Actividad:	A2P6.4.5	Actividad:	Administrar el diseño
Objetivo de la actividad:	Evaluar periódicamente las características de diseño en caso de evolución del sistema de software y de su arquitectura, así como pronosticar la posible obsolescencia de componentes y tecnologías, su reemplazo por otros a lo largo del tiempo en el ciclo de vida del sistema de software.				
Entradas:	Registro del diseño del Sistema				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Organizar y controlar de los modelos y vistas de la arquitectura para ayudar a garantizar que se cumpla la intención arquitectónica.			Analista de sistemas	Formato de diseño del sistema (Anexo 15)
2	Es necesario evaluar la arquitectura de software existen varias técnicas tanto cualitativas (escenarios, cuestionarios o listas de verificación) y cuantitativas (métricas, simulaciones, prototipos, modelos matemáticos).				
	Informar sobre la evolución tecnológica y mejore el sistema de software si fuese necesario.				
3	Mantener la trazabilidad entre las interfaces y los casos de usos de los requerimientos funcionales, y estos a su vez con sus respectivos diagramas de vistas de arquitectura.				
Salidas	Registro de Diseño del sistema				

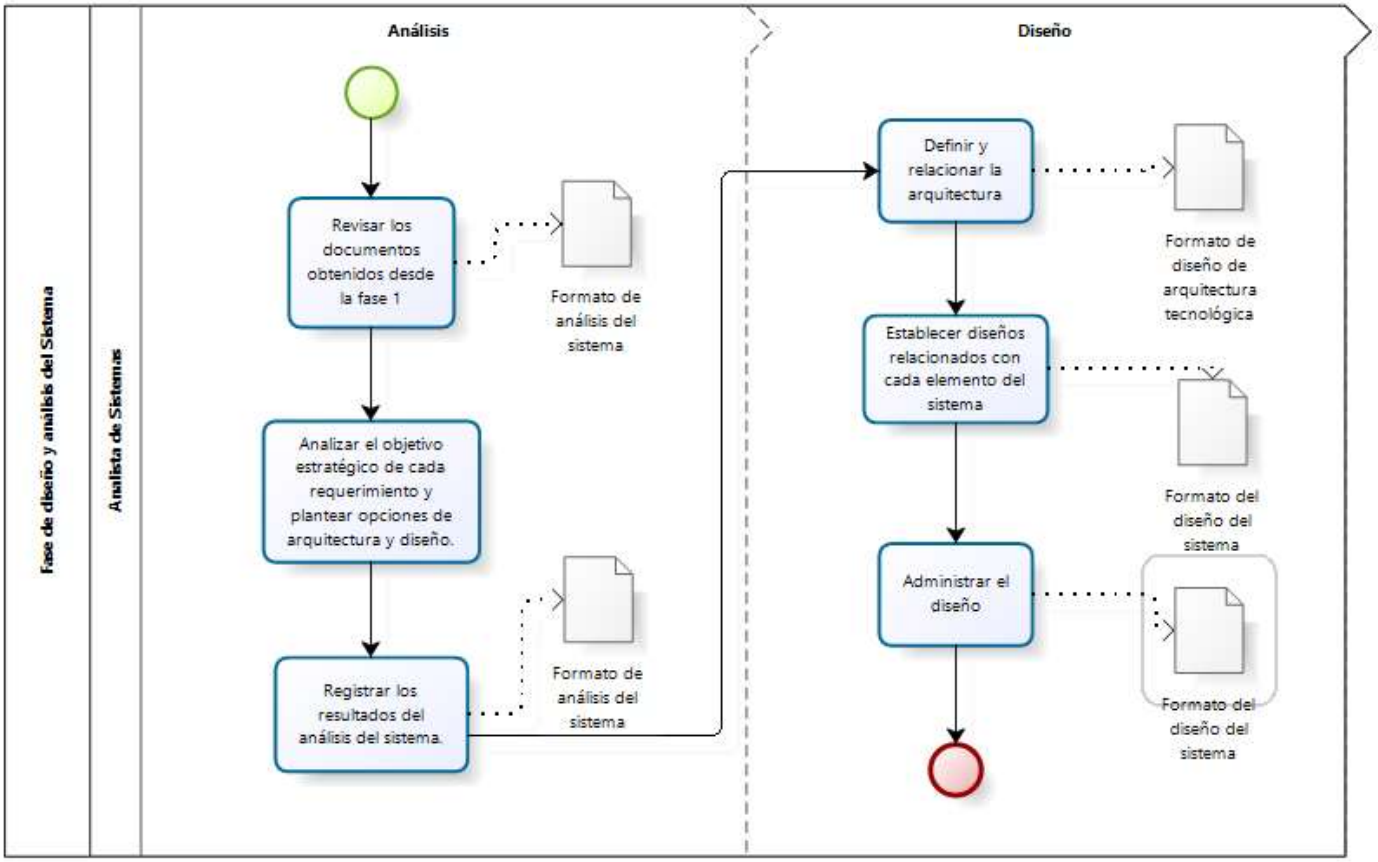


Ilustración 20: Diagrama – Diseño y análisis de sistemas
Fuente: Elaboración propia

4.1.3. Fase de implementación y pruebas de unidades

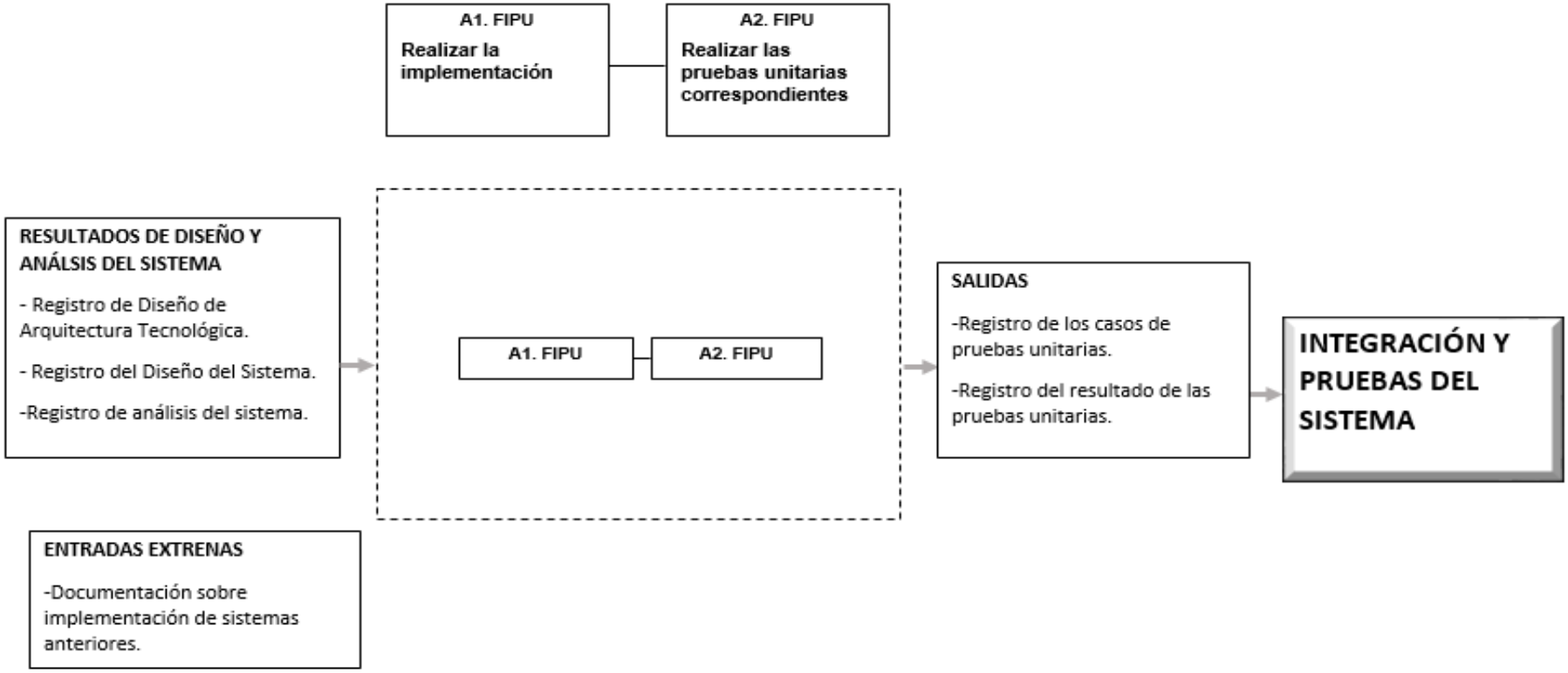


Ilustración 21: Fase de implementación y pruebas de unidades
Fuente: Elaboración propia

Proceso 6.4.7 ISO 12207:2017	Proceso de implementación	Código de la Actividad:	A1P6.4.7	Actividad:	Realizar la implementación
Objetivo de la actividad:	Llevar a cabo la implementación del software y realizar las pruebas unitarias necesarias.				
Entradas:	Registro de análisis del sistema				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Transformar la especificación de RQ y diseño del sistema en código ejecutable.			Analista de calidad y Desarrollador	Formato de Plan de pruebas (Anexo 20)
2	Codificar y estructurar clases y objetivos en términos de componentes (código fuente, ejecutables, bases de datos, etc.)				
3	Definir la organización del código en términos de subsistemas estructurados en capas.				
4	Elaborar casos de pruebas unitarias				
5	Ejecutar pruebas unitarias				
6	Registrar evidencia objetiva de que el elemento del sistema de software cumple con los requisitos y de los errores encontrados si fuera el caso				
Salidas	Registro del resultado de las pruebas unitarias				

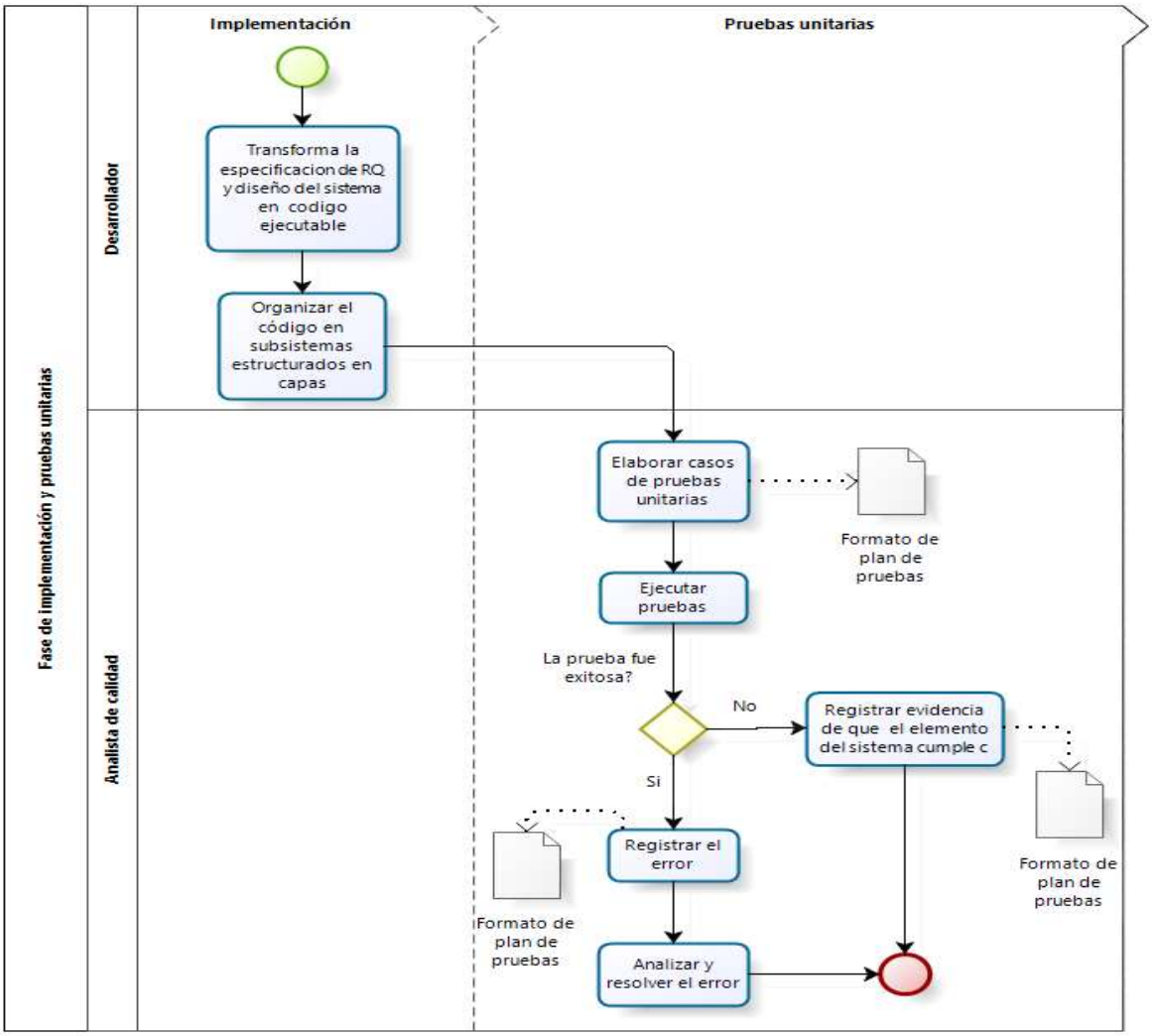


Ilustración 22: Diagrama – Implementación y pruebas unitarias
Fuente: Elaboración propia

4.1.4. Fase de integración y pruebas del sistema

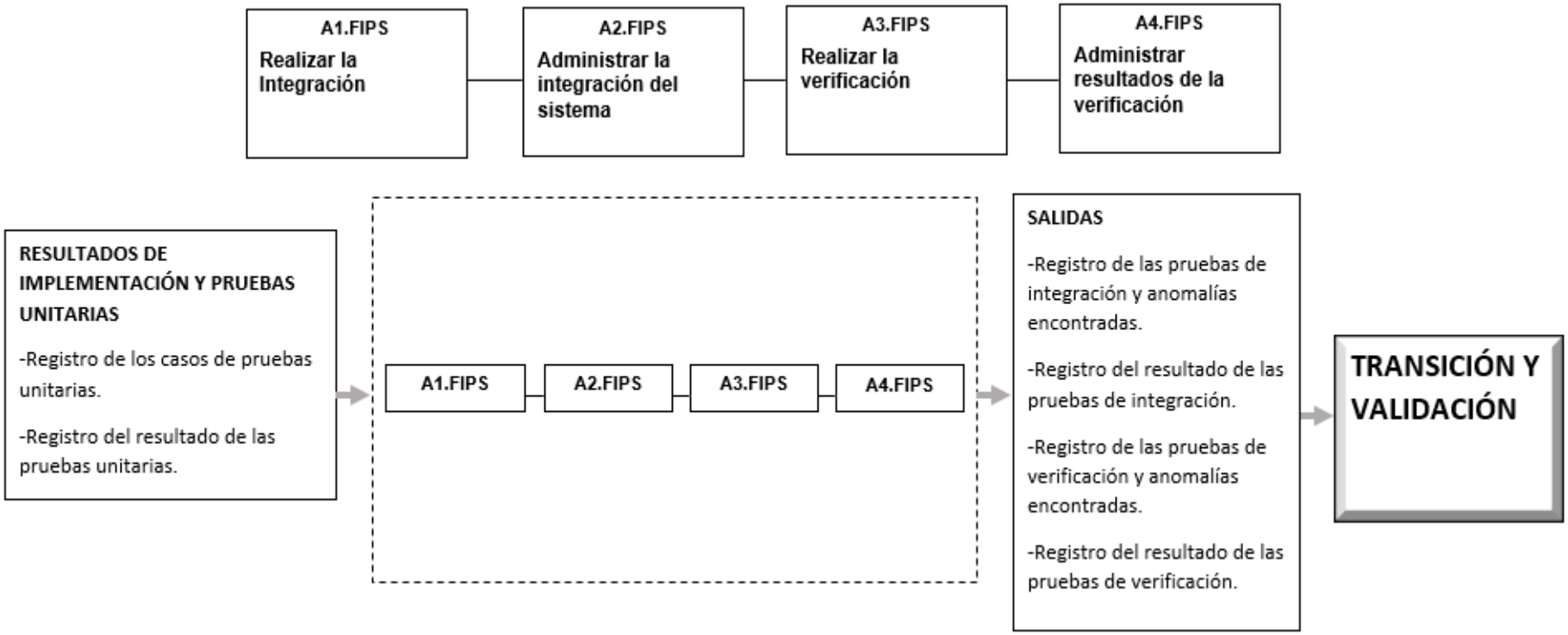


Ilustración 23: Fase de integración y pruebas del sistema
Fuente: Elaboración propia

Proceso 6.4.8 ISO 12207:2017	Proceso de integración	Código de la Actividad:	A1P6.4.8	Actividad:	Realizar la integración
Objetivo de la actividad	El propósito del proceso de integración es sintetizar un conjunto de elementos del sistema en un sistema realizado (producto o servicio) que satisface los requisitos del sistema, la arquitectura y el diseño.				
Entradas	Registro de la estrategia de integración. Registro del resultado de las pruebas unitarias				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Integrar las unidades del sistema con el diseño y los requerimientos.			Analista de calidad y desarrollador	Formato de Plan de pruebas (Anexo 20)
2	Realizar un plan de pruebas				
3	Realizar las pruebas de integración (pruebas del funcionamiento de los diferentes módulos de manera agrupada).				
4	Realizar las pruebas del sistema				
Salidas	Registro de las pruebas de integración y anomalías encontradas.				

Proceso 6.4.8 ISO 12207:2017	Proceso de integración	Código de la Actividad:	A2P6.4.8	Actividad:	Administrar los resultados de la integración
Objetivo de la actividad:	Registrar resultados de la integración; y mantener la trazabilidad bidireccional entre los elementos del sistema integrado y la arquitectura del sistema de software, el diseño y los requisitos del sistema.				
Entradas:	Registro del resultado de las pruebas de integración y anomalías encontradas.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Registrar resultados de integración y las anomalías encontradas.			Analista de calidad y desarrollador	Formato de Plan de pruebas (Anexo 20)
2	Corregir errores y ver que el software cumpla con los requerimientos funcionales y no funcionales.				
3	Realizar la trazabilidad de los elementos del sistema de software integrado.				
4	Mantener la trazabilidad entre el resultado obtenido y el resultado esperado de la corrección.				
Salidas	Registro del resultado de las pruebas de integración.				

Proceso 6.4.9 ISO 12207:2017	Proceso de verificación	Código de la Actividad:	A1P6.4.9	Actividad:	Realizar la verificación
Objetivo de la actividad:	El propósito del proceso de verificación es proporcionar evidencia objetiva de que un sistema o elemento del sistema cumple con los requisitos y características especificadas.				
Entradas:	Registro del resultado de las pruebas de integración.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Realizar la verificación a los elementos de software que presentan errores.			Analista de calidad y desarrollador	Formato de plan de pruebas (Anexo 20)
2	corregir errores y ver que el software cumpla con los requerimientos funcionales y no funcionales.				
Salidas	Registro de las pruebas de verificación y anomalías encontradas.				

Proceso 6.4.9 ISO 12207:2017	Proceso de verificación	Código de la Actividad:	A2P6.4.9	Actividad:	Administrar los resultados de la verificación
Objetivo de la actividad:	Revisar los resultados de verificación y las anomalías encontradas; así como registrar incidentes y problemas. Obtener el acuerdo de partes interesadas de que el sistema o elemento de software cumple con los requisitos especificados.				
Entradas:	Registro del resultado de las pruebas de verificación y anomalías encontradas.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Registrar incidentes y problemas durante la verificación y realice un seguimiento de su resolución.			Analista de calidad y desarrollador	Formato de plan de pruebas (Anexo 20)
2	Revisar los resultados de verificación y el grado de corrección.				
3	Mantener la trazabilidad entre el resultado obtenido y el resultado esperado de la corrección.				
Salidas	Registro del resultado de las pruebas de verificación.				

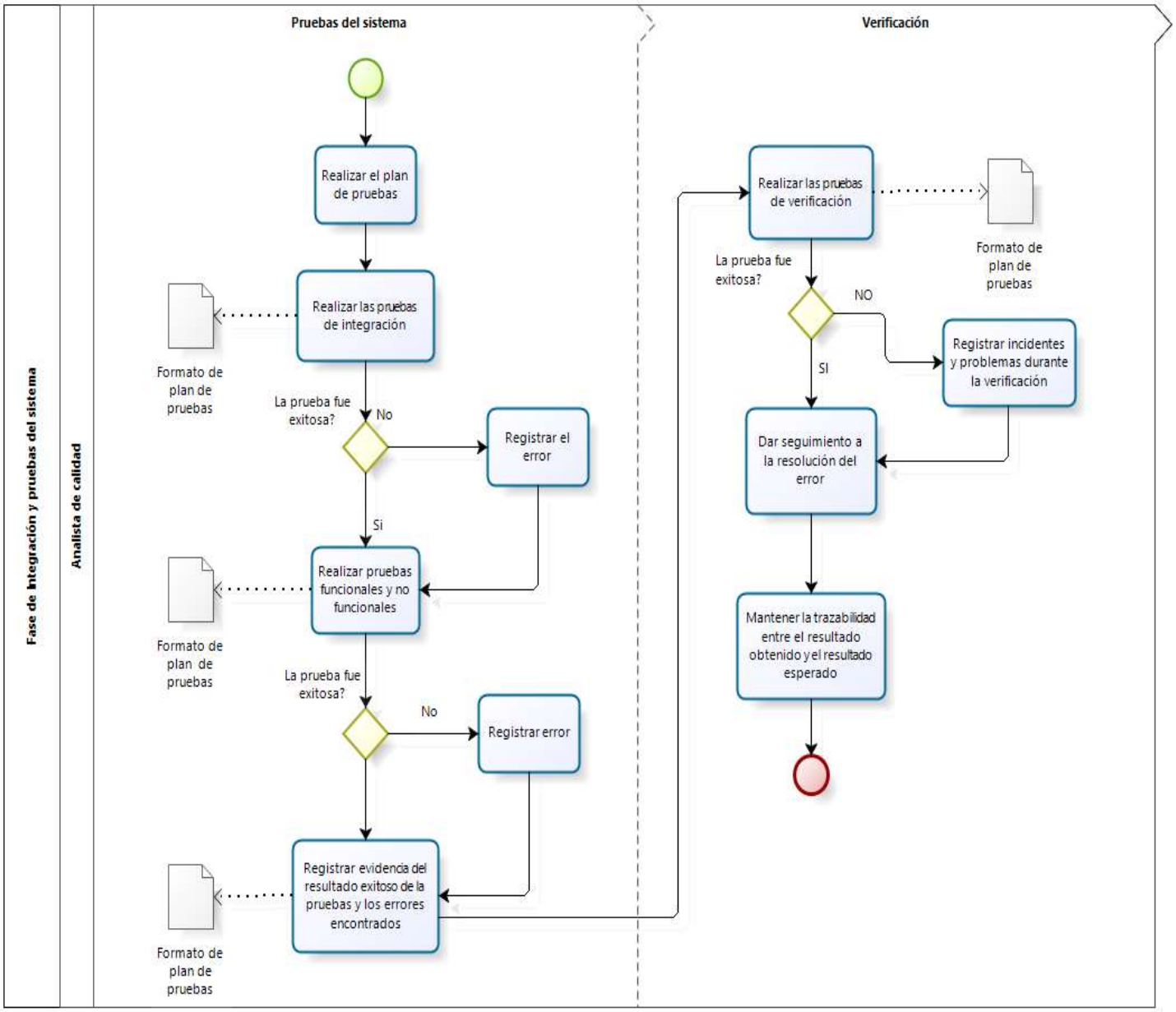


Ilustración 24: Diagrama – Integración y pruebas del sistema
Fuente: Elaboración propia

4.1.5. Fase de transición y validación

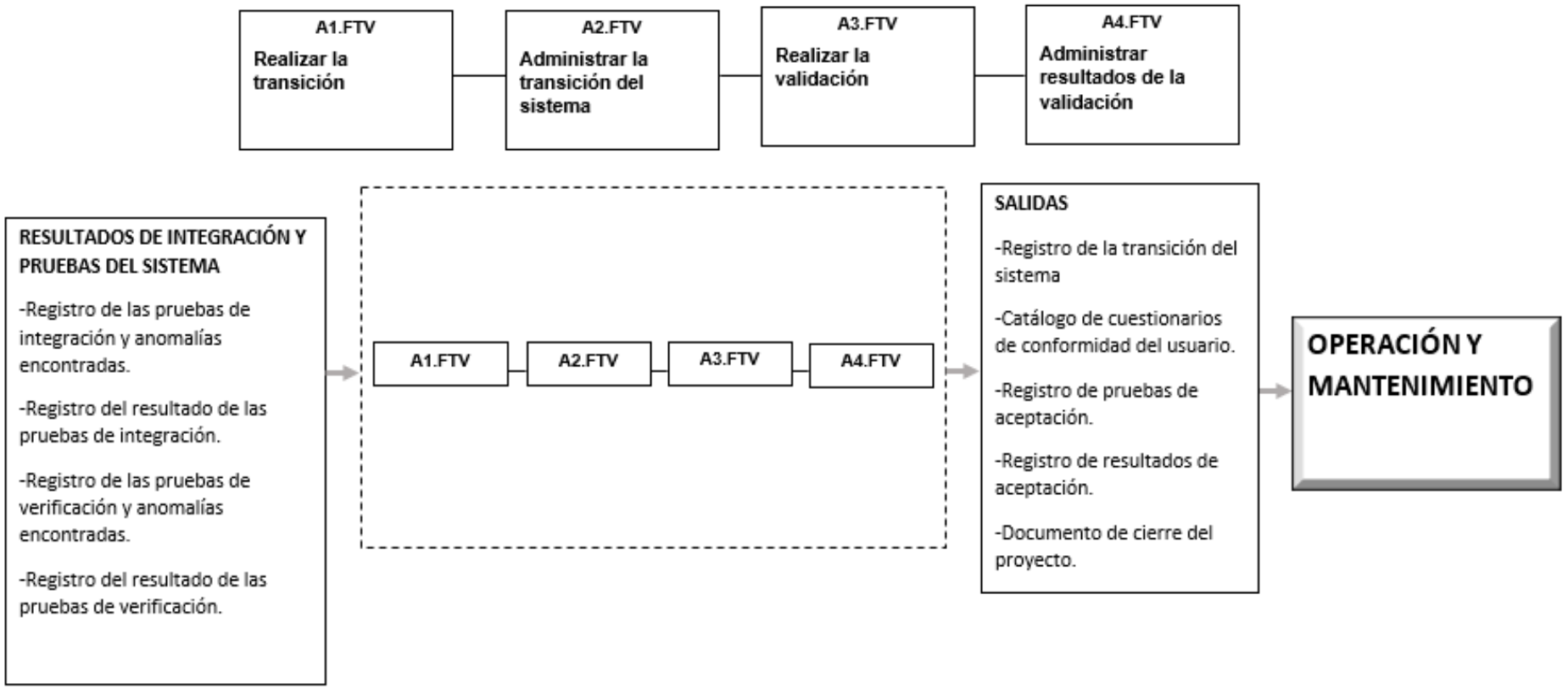


Ilustración 25: Fase de transición y validación
Fuente: Elaboración propia

Proceso 6.4.10 ISO 12207:2017	Proceso de transición	Código de la Actividad:	A1P6.4.10	Actividad:	Realizar la transición
Objetivo de la actividad:	Preparar el sitio de operación de acuerdo con los requisitos de instalación; así como entregar el sistema para la instalación en el lugar. De la misma manera proporcionar capacitación para los usuarios.				
Entradas:	Registro del resultado de las pruebas de verificación.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Preparar el sitio de operación o el entorno virtual de acuerdo con los requisitos de instalación.			Analista de Sistemas	Formato de implantación del sistema (Anexo 21)
2	Entregar el sistema para la instalación en el lugar y la hora correctos.				Formato de capacitación (Anexo 19)
3	Instalar el producto en su ubicación operativa física o virtual y con su entorno.				
4	Proporcionar documentación y capacitación para los usuarios.				
5	Agregar tutoriales, manuales interactivos para los usuarios. (Anexo 23 – Obs.4)				
Salidas	Registro de la instalación del sistema.				

Proceso 6.4.11 ISO 12207:2017	Proceso de validación de software	Código de la Actividad:	A1P6.4.11	Actividad:	Realizar la validación
Objetivo de la actividad:	Proporcionar evidencia objetiva de que el sistema, cuando está en uso, cumple con requisitos de las partes interesadas, logrando su uso previsto.				
Entradas:	Registro de especificación de requerimientos				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Realizar las pruebas de aceptación finales (pruebas de interfaz y funcionamiento).			Analista de calidad y desarrollador	Formato de Plan de pruebas (Anexo 20) Formato de Acta de cierre (Anexo 22)
2	Registrar el resultado de la validación.				
3	Resolver los incidentes encontrados durante la validación.				
4	Realizar una reunión conmemorativa de cierre de proyecto.				
Salidas	Registro de las pruebas de aceptación. Acuerdo de cierre de proyecto				

Proceso 6.4.11 ISO 12207:2017	Proceso de validación de software	Código de la Actividad:	A2P6.4.11	Actividad:	Administrar los resultados de la validación
Objetivo de la actividad:	Revisar los resultados de validación y las anomalías encontradas; así como registrar incidentes y problemas. Obtener el acuerdo de parte del usuario de que el sistema cumple con los requisitos especificados.				
Entradas:	Registro del resultado de las pruebas de aceptación. Formato de cierre de proyecto				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Registrar los resultados de validación y las anomalías encontradas.			Analista de Calidad y desarrollador	Formato de Plan de pruebas (Anexo 20)
2	Registrar la solución de incidentes encontrados durante la validación.				
3	Mantener la trazabilidad de los elementos del sistema validados.				
4	Obtener el acuerdo de las partes interesadas de que el sistema cumple con las necesidades de las partes interesadas.				
Salidas	Registro del resultado de las pruebas de aceptación.				

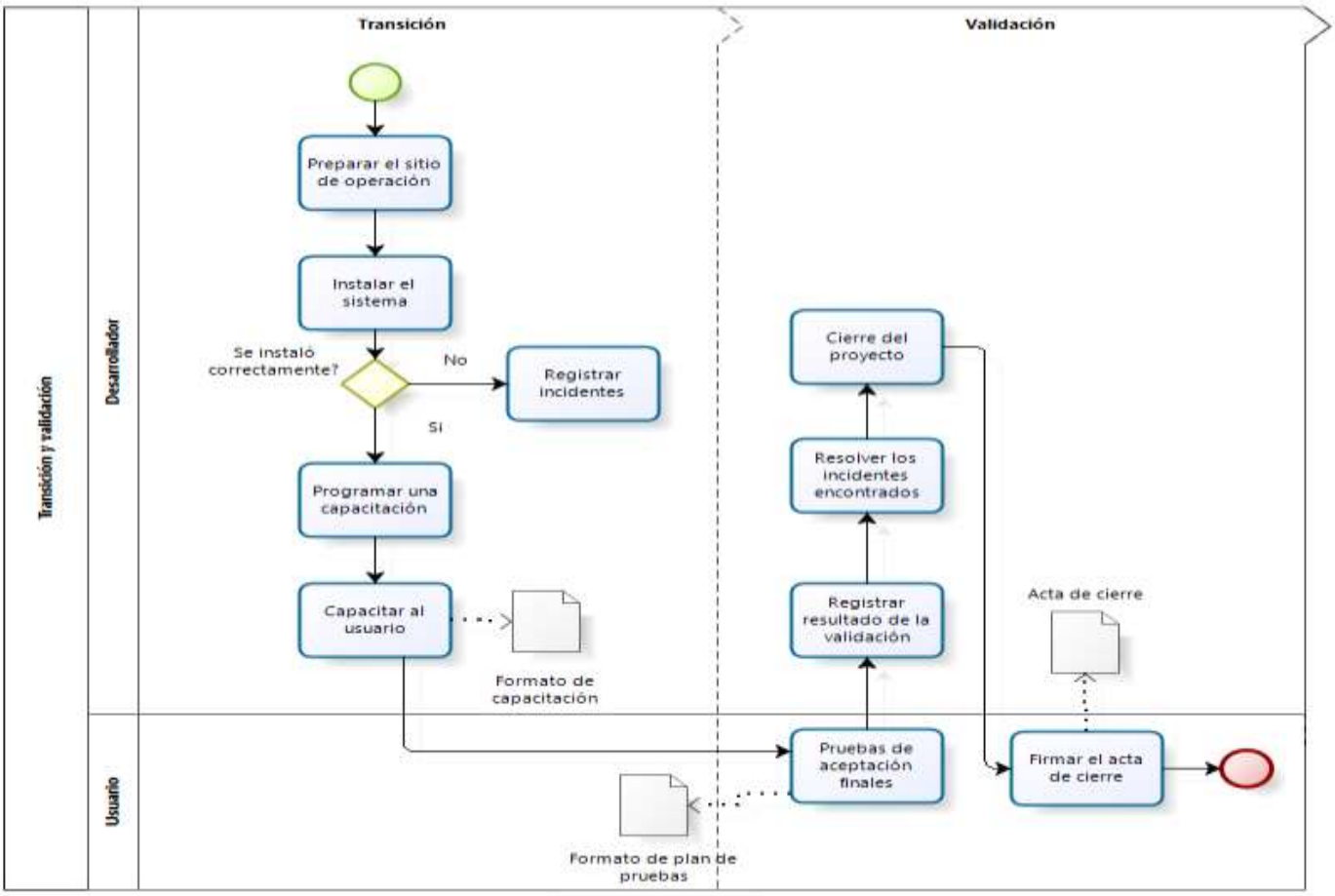


Ilustración 26: Diagrama – Transición y validación
Fuente: Elaboración propia

4.1.6. Fase de operación y mantenimiento

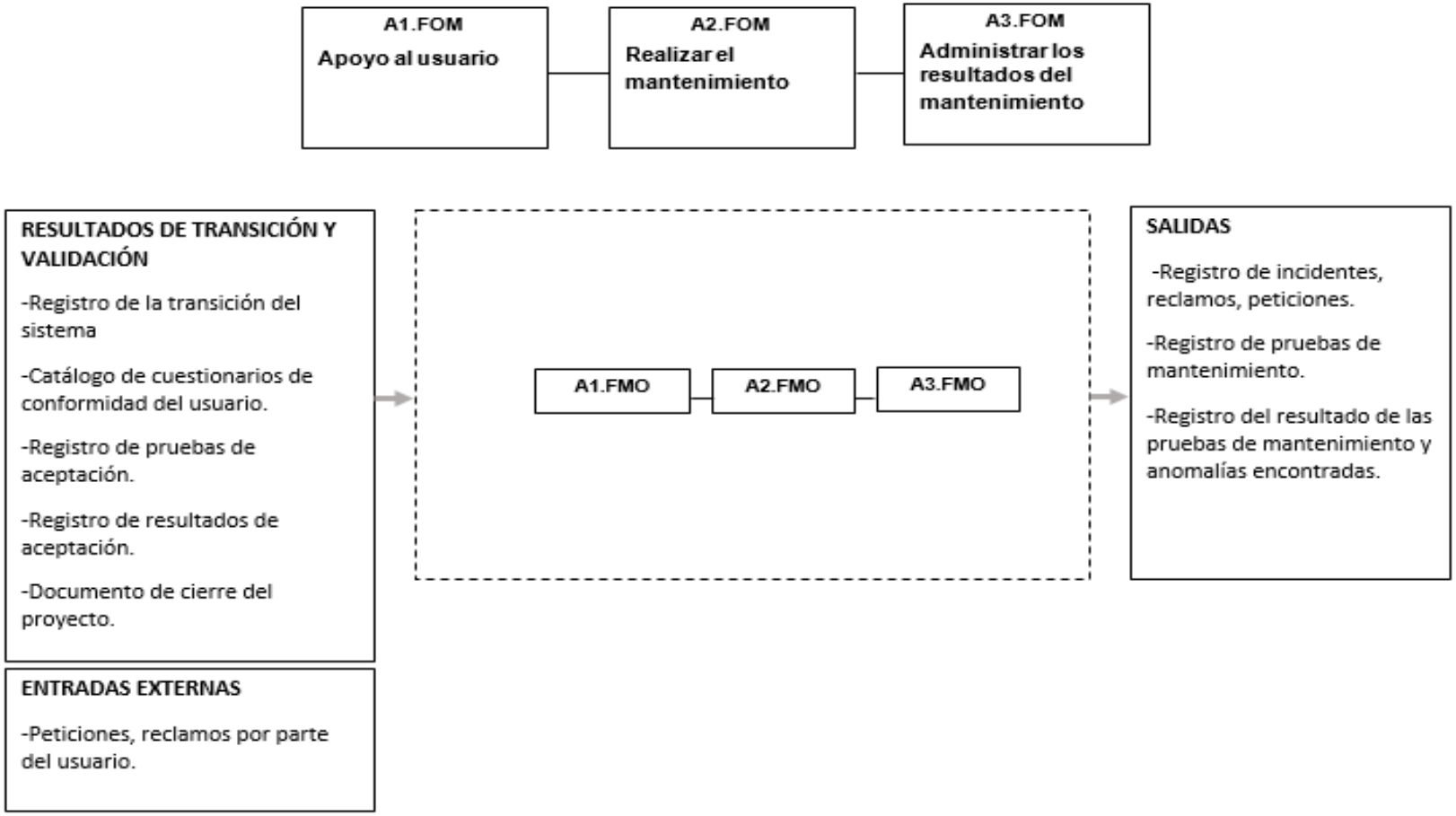


Ilustración 27: Fase de operación y mantenimiento
Fuente: Elaboración propia

Proceso 6.4.12 ISO 12207:2017	Proceso de operación	Código de la Actividad:	A1P6.4.12	Actividad:	Apoyar al usuario
Objetivo de la actividad:	Proporcionar asistencia y consultar a los clientes y usuarios para resolver quejas, incidentes, problemas y solicitudes de servicio; así como determinar el grado en que el sistema o los servicios de software entregados satisfacen las necesidades de los usuarios.				
Entradas:	Registro de incidentes, reclamos, peticiones.				
	Registro del resultado de las pruebas de validación.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Registrar la petición de cambio o modificación con motivo de la detección de un problema o por la necesidad de una mejora.			Analista de Sistemas	Formato de incidentes y reclamos. (Anexo 18)
2	Analizar el alcance de la petición en lo referente a los sistemas de información afectados, valorando hasta qué punto pueden ser modificados en función del ciclo de vida estimado				
3	Proporcionar asistencia y consultar a los usuarios para resolver quejas, incidentes, problemas y solicitudes de servicio.				
4	Determinar el grado en que el sistema satisfacen las necesidades de los clientes y usuarios.				
Salidas	Registro de Incidentes resueltos y pendientes.				

Proceso 6.4.13 ISO 12207:2017:	Proceso de mantenimiento del software	Código de la Actividad:	A1P6.4.13	Actividad:	Realizar el mantenimiento
Objetivo de la actividad:	Identificar las necesidades correctivas y de mantenimiento preventivo. Analizar el impacto de los cambios de mantenimiento Identificar cuándo se requiere mantenimiento adaptativo o perfectivo.				
Entradas:	Registro de Incidentes resueltos y pendientes.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Revisar las quejas, los eventos, los incidentes y los informes de problemas para identificar qué tipo de mantenimiento se aplicará (correctivo, adaptativo, perfectivo y preventivo).			Desarrollador	Formato de mantenimiento del sistema (Anexo 17)
2	Analizar el impacto de los cambios de mantenimiento en las estructuras de datos, los datos y las funciones de software relacionadas, la documentación del usuario y las interfaces.				
3	Implementar los procedimientos para la corrección de defectos y errores, o para el reemplazo o actualización de los elementos del sistema.				
Salidas	Registro de la aplicación del mantenimiento al sistema				

Proceso 6.4.13 ISO 12207:2017	Proceso de mantenimiento del software	Código de la Actividad	A2P6.4.13	Actividad	Administrar los resultados del mantenimiento
Objetivo de la actividad:	Registrar incidentes y problemas; así como las tendencias de incidentes, problemas y acciones de mantenimiento. Monitorear y medir la satisfacción del usuario con el sistema.				
Entradas:	Registro del resultado de las pruebas de mantenimiento y anomalías encontradas.				
N°	**Tareas**			**Responsable**	**Artefactos**
1	Registrar incidentes y problemas, incluidas sus resoluciones, y resultados significativos de mantenimiento.			Desarrollador	Formato de mantenimiento del sistema. (Anexo 17)
2	Identificar y registrar las tendencias de incidentes, problemas y acciones de mantenimiento.				
3	Mantener la trazabilidad de los elementos del sistema que se le han dado mantenimiento.				
4	Monitorear y medir la satisfacción del cliente con el sistema y soporte de mantenimiento.				
Salidas	Registro del resultado del mantenimiento del sistema				

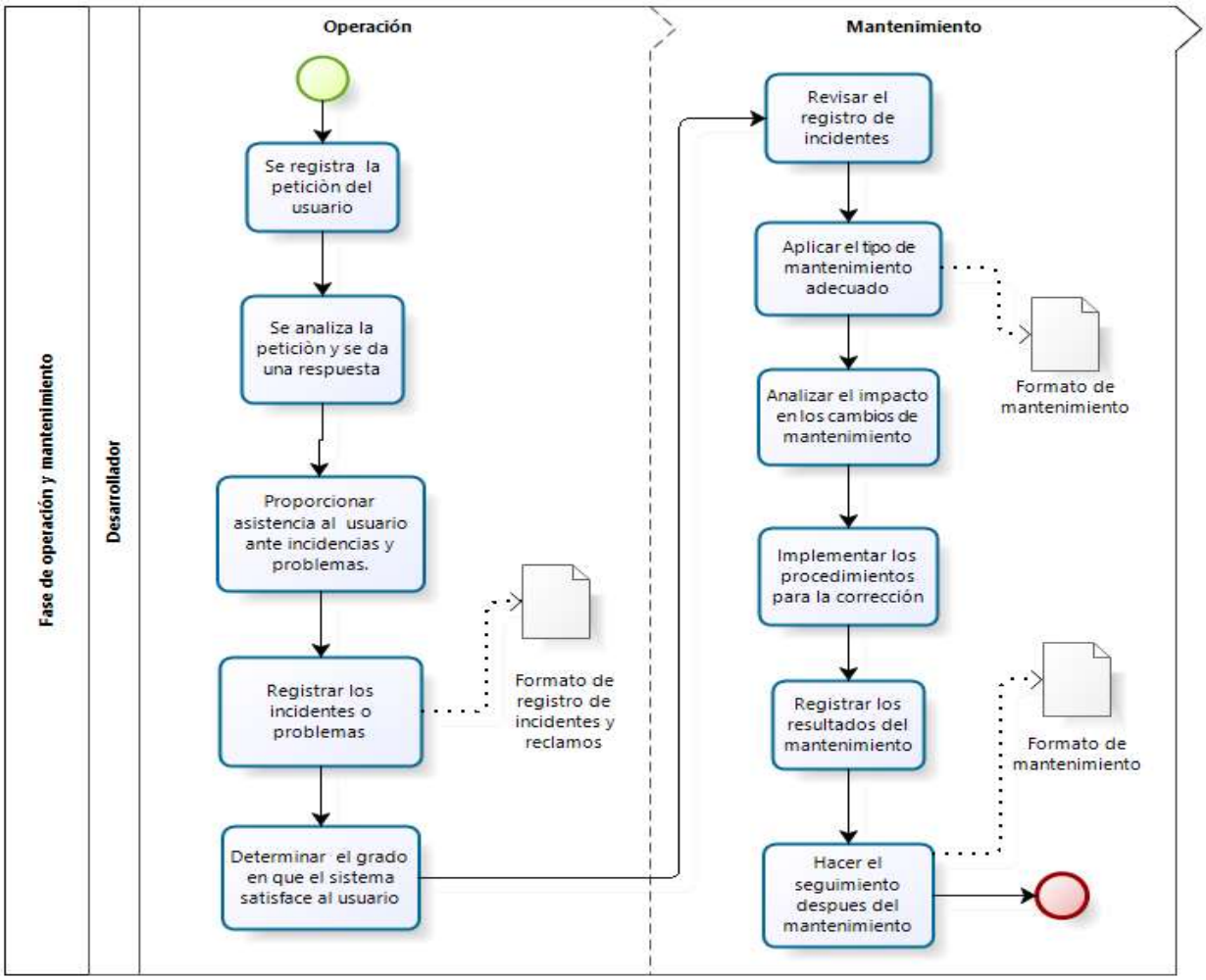


Ilustración 28: Diagrama – Operación y mantenimiento
Fuente: Elaboración propia

4.1.7. METODO DE ESTIMACIÓN DE PUNTOS DE CASOS DE USO

Es un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de pesos a un cierto número de factores que lo afectan. Este método realiza la estimación en base a casos de uso.

En el modelo que se está proponiendo a el área de desarrollo, tendrá que realizar casos de uso para plasmar los requerimientos del usuario, por lo que este método de estimación sería el más adecuado para la MPCH.

Ventajas:

- ✓ Para el área de desarrollo el método es fácil de entender y, a su vez, fácil de aplicar, para los trabajadores.
- ✓ Fácil adaptación para la MPCH ya que se utiliza la técnica de los casos de uso.
- ✓ Ofrece la posibilidad de estimar las horas-hombre de un proyecto de software que esté desarrollando la MPCH.
- ✓ Puede utilizarse para estimar las horas- hombre en las pruebas que se realicen con casos de uso.

PROCEDIMIENTO DE CÁLCULO DE LOS PUNTOS DE CASOS DE USO DEL SISTEMA QUE SE VA A DESARROLLAR

A continuación, se van a explicar los pasos generales para el cálculo de puntos casos de uso.

(a) Calcular los puntos casos de uso sin ajustar:

Para realizar el cálculo se tiene que realizar los siguientes pasos:

Clasificar cada interacción entre actor y caso de uso según su complejidad y asignar un peso en función de ésta (**UAW**).

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema que se va a desarrollar y la complejidad de cada uno de ellos. La complejidad de los actores se establece, teniendo en cuenta en primer lugar, si se trata de un usuario o de otro sistema de la MPCH, y, en segundo lugar, la forma en que el actor interactúa con el sistema.

Tipo de interacción	Descripción	Factor de peso	N° de actores	Resultado
Simple	Cualquier sistema de la MPCH que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Aplication Programming Interface)	1	A	1*A
Promedio	Cualquier sistema de la MPCH que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	B	2*B
Complejo	El usuario que interactúa con el sistema mediante una interfaz gráfica.	3	C	3*C
			Total UAW=	**A+2B+3C**

- Calcular la complejidad de caso de uso del sistema que se va a desarrollar, según número de transacciones o pasos del mismo: (**UUCW**)

Tipo de Caso de Uso	Descripción	Factor de peso	N° de Caso de Uso	Resultado
Simple	1 – 3 transacciones	5	A	5*A
Promedio	4 – 7 transacciones	10	B	10*B
Complejo	Mayor de 8 transacciones	15	C	15*C
			Total UUCW=	**5A+10B+15C**

- Calcular los puntos casos de uso sin ajustar del sistema que se va a desarrollar.
  Para obtener los puntos de casos de usos sin ajustar utilizaremos la siguiente fórmula:

UUCP = UAW + UUCW

(b) Calcular los puntos casos de uso ajustados:

- Cálculo de los factores técnicos (**TCF**)

  Este coeficiente se calcula mediante la cuantificación de un conjunto de 13 factores que determinan la complejidad técnica del sistema va a desarrollar la MPCH. Cada uno de los factores tiene un peso definido y cada factor se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante.

Número de factor	Descripción	Comentario del grupo involucrado	Peso (W)	Valor (V)	Factor (W*V)
T1	Sistema Distribuido				
T2	Tiempo de respuesta				

T3	Eficiencia por el usuario				
T4	Proceso interno complejo				
T5	Reusabilidad				
T6	Facilidad de instalación				
T7	Facilidad de uso				
T8	Portabilidad				
T9	Facilidad de cambio				
T10	Concurrencia				
T11	Objetivos especiales de seguridad				
T12	Acceso directo a terceras partes				
T13	Facilidades especiales de entrenamiento a usuarios finales				
			Total =		$\sum(W*V)$

Con la siguiente fórmula se calcula el factor técnico que aplicaremos en la fórmula final para calcular los puntos de casos de uso ajustados:

$$\mathbf{TCF = 0.6 + (0.01 * \sum(W*V))}$$

- Cálculo de los factores de entorno (**EF**)

  Las habilidades y el entrenamiento de los trabajadores del área de desarrollo de la MPCH, tienen un gran impacto en las estimaciones de tiempo. Estos son 8 factores los que se contemplan en el cálculo del Factor de ambiente. Cada uno de los factores tiene un peso definido y cada factor se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante.

Número del factor	**Descripción**	**Comentario del grupo involucrado**	**Peso (W)**	**Valor (V)**	**Factor (W*V)**
E1	Familiaridad con el modelo del proyecto usado.		1.5		
E2	Experiencia en la aplicación		0.5		
E3	Experiencia OO.		1		
E4	Capacidad del analista líder.		0.5		

E5	Motivación.		1		
E6	Estabilidad de los requerimientos.		2		
E7	Personal media jornada.		-1		
E8	Dificultad en lenguaje de programación.		-1		
			Total =		$\sum(W * V)$

Con la siguiente fórmula se calcula el factor entorno que aplicaremos en la fórmula final para calcular los puntos de casos de uso ajustados:

$$\mathbf{EF = 1.4 + (0.03 * \sum(W * V))}$$

- Calcular los puntos casos de uso sin ajustar

  Para obtener los puntos de casos de usos ajustados utilizaremos la siguiente fórmula:

$$\mathbf{UCP = UUCP * TCF * EF}$$

PROCEDIMIENTO DE CÁLCULO DEL ESFUERZO

Este cálculo se realiza con el fin de tener una aproximación del esfuerzo, pensando solo en el desarrollo según las funcionalidades de los casos de uso. Para ello se utilizará la siguiente fórmula:

El esfuerzo en horas-hombre viene dado por:

$$\mathbf{E = UCP * PF}$$

E: esfuerzo estimado en horas-hombre.

UCP: Puntos de casos de uso ajustados.

PF: Factor de conversión (20 horas-hombre por defecto).

Se debe tener en cuenta que este método proporciona una estimación del esfuerzo en horas-hombre contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso.

Para la obtención de una estimación más exacta de la duración del proyecto, se hace necesario agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las restantes actividades que se llevaron a cabo durante el desarrollo del software; así se la distribución del esfuerzo entre dichas actividades está dada por la siguiente aproximación:

Distribución genérica del esfuerzo.

Actividad	Porcentaje
Análisis	10.00%
Diseño	20.00%
Programación	40.00%
Pruebas	15.00%
Sobrecarga(otras actividades)	15.00%

Con este criterio y tomando como entrada la estimación de tiempo calculada a partir de los Puntos de Casos de Uso, se pueden calcular las demás estimaciones para obtener la duración total del proyecto.

Distribución real del esfuerzo.

Actividad	**Horas-Hombre**
Análisis	
Diseño	
Programación	
Pruebas	
Sobrecarga (otras actividades)	
Total	

CAPÍTULO V: RESULTADOS y DISCUSIÓN DE RESULTADOS

5.1. Validación del modelo de aplicación basado en la norma ISO/IEC 12207, al proceso de desarrollo de software en la Municipalidad de Chiclayo

Aplicando el formato de encuesta que se muestra en el Anexo 22, se obtuvieron las valoraciones de cada uno de los expertos para cada uno de los criterios considerados para validar el modelo propuesto para el Proceso de desarrollo de software, cuyos resultados se muestran a continuación:

Tabla 28: Resultados de la validación de expertos del modelo propuesto para la Gestión del Desarrollo del Software, basada en la NTP ISO/IEC 12207:2017 en la Municipalidad Provincial de Chiclayo

FASES	PROCESOS	EXPERTO 1					EXPERTO 2					EXPERTO 3									
		FU	RE	FI	US	MA	FU	RE	FI	US	MA	FU	RE	FI	US	MA	FU	RE	FI	US	MA
Análisis y definición de requerimientos	Definición de requisitos y necesidades de las partes interesadas	4	3	4	4	4	3	3	4	3.5	3.5	3	3	3	3	3	3.3	3	3.7	3.6	3.4
	Definición de requisitos del Sistema de software	4	3	4	4	3.5	3	3	4	4	3.5	3	3	3	3	3					
Análisis y Diseño del sistema	Análisis del sistema	3	3	3	3	4	3	3	3	4	4	3	3	3	3	3.5	3.3	3.3	3.3	3.4	3.7
	Definición de arquitectura del software	4	4	4	4	3.5	4	4	4	3.5	4	3	3	3	3	4					
	Diseño del software	3	3	3	3	3.5	4	4	4	4	3.5	3	3	3	3	3					
Implementación y pruebas de unidades	Implementación del software	4	4	4	4	4	3	3	3	4	4	3	3	3	4	4	3.3	3.3	3.3	4	4
Integración y pruebas del sistema	Integración del software	4	3	4	3	4	3.5	3.5	3.5	4	4	3	3	3	3	3	3.4	3.3	3.5	3.4	3.6
	Verificación del software	3.5	4	4	4	4	3.5	3.5	3.5	3.5	3.5	3	3	3	3	3					
Transición y validación	Transición del software	3	4	4	3	3.5	3.5	4	3.5	4	4	3	3	3	3	4	3.3	3.6	3.6	3.4	3.7
	Validación del software	4	4	4	4	3.5	3.5	3.5	4	3.5	4	3	3	3	3	3					
Operación y mantenimiento	Operación del software	3.5	3.5	4	4	4	3	3	3	3.5	4	3	3	3	3	3	3.2	3.2	3.2	3.5	3.8
	Mantenimiento del software	4	4	3.5	3.5	3.5	3	3	3	4	4	3	3	3	3	4					

CAPITULO VI:

CONCLUSIONES

De las calificaciones, que los expertos dieron en su valoración de la metodología propuesta para la gestión de desarrollo de Software, aplicando los marcos de referencia ISO/IEC 12207:2017 para la Municipalidad Provincial de Chiclayo se realizaron las siguientes interpretaciones:

- Permite un control adecuado sobre las planificaciones que se deben de efectuar a nivel de tareas, actividades, recursos, entre otros, que permiten medir la calidad del software.
- La descripción de las actividades y tareas son claras y comprensibles en su explicación para su ejecución como parte de la metodología.
- Permite una verificación, validación y captura de información necesaria para dar cumplimiento a los atributos de calidad analizados.
- Las actividades y tareas desarrolladas en cada etapa son suficientes para lograr los objetivos o resultados esperados de la metodología.
- El uso de la metodología permite niveles aceptables de fiabilidad a los usuarios en el funcionamiento diario y operatividad en caso de fallos.
- El uso de la metodología permite llevar un mantenimiento adecuado de las aplicaciones que están en funcionamiento en la MPCH.
- El uso de la metodología permite alcanzar buen nivel de rendimiento que proporcione adecuados tiempos de respuesta y procesamiento de las aplicaciones de la MPCH.
- El uso de la metodología permite un nivel aceptable de usabilidad por parte de los usuarios y trabajadores del área de TI.

RECOMENDACIONES

- Se recomienda a la MPCH designar a una persona de la empresa que tenga como responsabilidad realizar las pruebas unitarias y de integración, para garantizar que los defectos no lleguen al usuario, asegurando así la máxima calidad del producto

- Continuar con el uso de herramientas y tecnologías actuales que tenga una base firme en buenas prácticas de desarrollo para ser aplicados en los futuros proyectos a fines o relacionados.

- Desarrollar una cultura de confianza sobre las aplicaciones solicitadas, esto será en la manera de integrar al usuario cada vez más en el control y desarrollo del proyecto, que permitirá mantener la comunicación sobre el desarrollo efectuado, así como de los resultados obtenidos.

- Desarrollar, permanentemente, actividades de capacitación que permitan conocer a los usuarios el flujo de la información y control que se realiza en las actividades del desarrollo de sus requerimientos solicitados.

Referencias bibliográficas

Adolfo, V. H. (26 de 02 de 2014). *SlideShare.* Obtenido de https://es.slideshare.net/adolfo0890/tarea-i-wong

Alarcón Aldana, A. C., González Sanabria, J. S., & Rodríguez Torres, S. L. (2011). Guía para pymes desarrolladoras de software, basada en la norma ISO/IEC 155041. *Revista Virtual Universidad Católica del Norte*, 29.

ANDALUCÍA, J. D. (2013). *Marco de desarrollo de la Junta de Andalucía*. Obtenido de http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/411

Anónimo. (-). *Codificacion del software*. Obtenido de hardware/software: https://4tesosite.wordpress.com/codificacion-del-software/

Anónimo. (09 de Septiembre de 2014). *Diagrama de UML.* Obtenido de Teatroabadia: https://www.teatroabadia.com/es/uploads/documentos/iagramas_del_uml.pdf

Aranda, R. A. (13 de 09 de 2019). *SCRIBD*. Obtenido de https://es.scribd.com/document/425652280/carta

Aybuke Aurum & Claes Wohlin. (2005). *Engineering and Managing Software Requirements.* Berlin: Spreinger.

Berzal, F. (30 de Noviembre de 2017). *Ciclo de Vida de un Sistema de Informacón.* Obtenido de http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf

BIBIÁN, O. P. (08 de 2017). PRUEBAS DE CALIDAD APLICADAS AL SITIO WEB ALLISON.

Canaan, R. (2017). *Lifeder.com*. Obtenido de https://www.lifeder.com/cultura-paracas/

Cantone, D. (2006). *Implementacion Y Debugging.* (C. A. Corp., Ed.)

Carazas, K. C. (2015). EVALUACIÓN DE LA CALIDAD DEL SISTEMA INTEGRAL DE RESTAURANTES-SIR, BASADO EN LA NORMA ISO/IEC 25000 DEL GRUPO UROS S.A.C. . Tacna, Perú.

Cillero, M. (2017). *Manuel Cillero*. Obtenido de https://manuel.cillero.es/doc/metrica-3/tecnicas/pruebas/sistema/

Corp, I. (2006). Obtenido de https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/deliveryprocesses/Transition_4EE85BB2.html_wpbs.html

Cueva, I. S. (10 de 2014). *Ingenieria de requisitos.* Ecuador.

Culoccioni, S. (25 de diciembre de 2014). *solvetic*. Obtenido de Programacion en 3 capas: https://www.solvetic.com/tutoriales/article/1378-programaci%C3%B3n-en-tres-capas-con-java/

Durand, S. W. (2017). *Análisis y requerimientos de software.* Huancayo.

El Emam, K., & Jung, H.-W. (2001). An empirical evaluation of the ISO/IEC 15504 assessment model. *The Journal of Systems and Software*, 19.

Fing. (s.f.). Obtenido de https://www.fing.edu.uy/inco/cursos/ingsoft/pis/proceso/MUM/dat/DimTiempo/faset.htm

Francisco Apodaca, F. A. (2012). *Meotodología de Software*. Obtenido de http://metodologiadesoftware.blogspot.com/2012/11/fases-del-modelo-rup_27.html

Galiano, F. B. (30 de Noviembre de 2017). *Refactorizacion y prueba de unidad.* Obtenido de elvex: http://elvex.ugr.es/decsai/csharp/design/nunit.xml

García, A. P. (2013). *Software de requerimientos.*

García, I. M. (Agosto de 2017). METODOLOGÍA PARA EL CÁLCULO DE COMPLEJIDAD EN PRUEBAS UNITARIAS DE CÓDIGO AUTOGENERADO. Querétaro.

García, M. & Garzás, J. (2008). La certificación por niveles de madurez de ISO/IEC 15504.

García, S. M. (2012). *Kybele.* Obtenido de Grupo de investigación: http://www.kybele.etsii.urjc.es/docencia/is4/2012-2013/material/is4.11.12.tema.xiii.mantenimientosw.pdf

Garzas, J. (s.f.). *sites*. Obtenido de La arquitectura de Software, el modelo 4+1: https://sites.google.com/site/jgarzas/4mas1

Garzás, J., Fernández, C. M., & Piattini, M. (2009). Una aplicación de ISO/IEC 15504 para la evaluación por niveles de madurez de PYMES y pequeños equipos de desarrollo. *Revista Española de Innovación, Calidad e Ingeniería de Software*, 14.

Gómez, V. (s.f.). *Arquitectura en tres capas*. Obtenido de Instinto Binario: https://instintobinario.com/arquitectura-en-tres-capas/

Grijalva, N. (2012). Obtenido de http://sofware1nathalygrijalva.blogspot.com/2012/10/modelo-espiral.html

Gutiérrez, A. R. (Septiembre de 2016). Desarrollo de APP para la realización de encuestas de movilidad.

Gutiérrez, P. (05 de Noviembre de 2013). *GENBETA*. Obtenido de Fundamento de las bases de datos: Modelo entidad-relación: https://www.genbeta.com/desarrollo/fundamento-de-las-bases-de-datos-modelo-entidad-relacion

Hernández, C. (2011). Obtenido de http://anaydisesistem.blogspot.com/2011/04/modelo-de-ciclo-de-vida.html

ICESI. (15 de Octubre de 2010). *ICESI*. Obtenido de Laboratorio de Ingerniería de Software Académica: http://www.icesi.edu.co/departamentos/tecnologias_informacion_comunicaciones/proyectos/lisa/home/analisis/diagramas_de_clases/inicio

IEEE. (2011). Ingeniería de sistemas y software: Requisitos y evaluación de calidad de sistemas y software. *ISO/IEC 25023*.

IEEE. (01 de diciembre de 2011). Procesos del ciclo de vida - Ingeniería de Requisitos. *ISO/IEC/29148*. SUIZA.

IEEE. (01 de diciembre de 2011). *Procesos del ciclo de vida - Ingeniería de Requisitos.* SUIZA.

IEEE. (Noviembre de 2017). Procesos del ciclo de vida del software-Ingeniería de sistemas y software. *ISO/IEC/12207*, 158. SUIZA.

ISO. (01 de 12 de 2011). Ingeniería de sistemas y software. Descripción de la arquitectura. *ISO/IEC/IEEE 42010*.

ITCA. (s.f.). *ITCA*. Obtenido de SELECCIÓN DE TÉCNICAS DE INGENIERÍA DE SOFTWARE: https://virtual.itca.edu.sv/Mediadores/stis/35___diseo_de_la_interfaz_de_usuario.html

Jan Bosch & Peter Molin. (1999). Software Architecture Design: Evaluation and Transformation. *Actas de la conferencia IEEE de 1999 sobre ingeniería de sistemas basados en computadoras* (pág. 10). ECBS'99.

José Martinez Guerero, Camilo Silva Delgado. (28 de Noviembre de 2010). *Guía Metodológica para el levantamiento y análisis de Requerimientos de Software en base a Procesos de Negocio.*

Karl Wiegers & Joy Beatty. (2013). *Software Requeriments.* Washintong: Microsoft.

Kenneth E. Kendall Y Julie E. Kendall. (2011). *ANALISIS Y DISEÑO DE SISTEMAS.* Mexico: Prentice Hall.

Kruchten, P. (1995). *Planos Arquitectónicos: El Modelo de "4+1" Vistas de la Arquitectura del Software.*

Landeta.P. (21 de Octubre de 2013). *Arquitectura del Software.* Obtenido de Sideshare: https://es.slideshare.net/landeta_p/2-1-vistas-arquitectonicas

Lozada, R. A. (2017). *GUÍA METODOLÓGICA PARA LA GESTIÓN DE REQUERIMIENTOS DE DESARROLLO DE SOFTWARE DEL GOBIERNO AUTÓNOMO DESCENTRALIZADO MUNICIPAL DE SAN MIGUEL DE IBARRA.* Ecuador.

Maldonado, G. (05 de Mayo de 2015). Análisis y Diseño de Sistemas. *Diagramas UML*.

Marichelo. (2016). Obtenido de http://marich.blogspot.es/1459223366/modelo-incremental/

Medina, M. (2003). Propuesta para un sistema de control de presupuestos en una Entidad Superior. Mexico D.F.

Méndez Nava , M. (Julio de 2006). Modelo de Evaluacion de Metodologias para El Desarrollo de Software. Caracas.

Mora, J. T. (2011). *Arquitectura de software para aplicaciones Web.* Mexico.

Morillo, R. A. (12 de Septiembre de 2012). *Ambientes de pruebas integrales de software: Buenas prácticas*. Obtenido de PmoInfórmática: http://www.pmoinformatica.com/2012/09/pruebas-software-ambientes.html

Morris, E. (08 de marzo de 2011). *Evaluación de software para la empresa*. Obtenido de https://www.esan.edu.pe/conexion/bloggers/el-blog-de-eddie-morris/2011/03/evaluacion-de-software-para-la-empresa-la-decision-correcta/

MPCH, P. (13 de 10 de 2010). *MANUAL DE PROCESOS - GERENCIA DE SISTEMAS E INFORMÁTICA.* Obtenido de munichiclayo: https://www.munichiclayo.gob.pe/index.php?tipo=doc&docT=Proc.%20Interno%20&docR=Documentos/documento0002612.pdf

NORAMBUENA, L. E. (2016). ANÁLISIS Y EVALUACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE PARA EL ÁREA DE CONSULTORÍA DE LA EMPRESA POWERDATA CHILE. Santiago de Chile, Chile.

OK HOSTING. (Revisado Enero 2018). *OK HOSTING.* Obtenido de Metodologías del Desarrollo del Software: https://okhosting.com/blog/metodologias-del-desarrollo-de-software/

Palacios, E. M. (Julio de 2015). MODELO DE PROCESOS BASADOS EN LA ISO/IEC 12207:2008 PARA LA ADQUISICIÓN DE SOFTWARE EN EL CENTRO VACACIONAL HUAMPAN. Trujillo.

Pietro Álvarez, C. G. (15 de Septiembre de 2015). Adaptación de las Metodologías Tradicionales Cascada y Espiral para la Inclusión de Evaluación Inicial de Usabilidad en el Desarrollo de Productos de Software en México.". Mexico.

Pino, Garcia, Ruiz, Piattini. (2006). Adaptación de las Normas ISO/IEC 12207:2002. *IEEE*, 8.

Pressman, R. .. (2010). *Ingeniería del software.* Mexico: McGraw-Hill.

Roberth. (2012). *blogspot*. Obtenido de http://ciclodevidasoft.blogspot.com/2012/07/modelos.html

Rodríguez, M. (25 de 04 de 2015). *Slideshare*. Obtenido de Slideshare: https://es.slideshare.net/AlarcosQualityCenter/promedesoft-2015-evaluacin-y-certificacin-de-la-calidad-del-producto-software-con-isoiec-25000

Roldan, J. L. (23 de Noviembre de 2010). *Pruebas Unitarias.* Obtenido de Slideshare: https://es.slideshare.net/ocomur/pruebas-unitarias-9368721

Shirley. (03 de 05 de 2012). *Ingenieria de Sistemas*. Obtenido de Tipos de diagrmas UML: http://ingenieriadesistemas-shirley.blogspot.com/2012/05/tipos-de-diagramas-uml.html

SINCOWS. (2013). *SINCOWS*. Obtenido de http://www.sincows.com/sincows/index.php?option=com_content&view=article&id=70&Itemid=68

Somerville, I. (2005). *Ingeniería del Software.* Madrid: Pearson Educacion S.A.

Sommerville, I. (2011). *Ingeniería de Software.* Mexico: Pearson.

Taramuel, L. A. (29 de Abril de 2011). Desarrollo e implementación de un aplicativo web para la gestion de concursos de la Asociacion de caballos de paso utilizando patrones de diseño MVC. loja, Ecuador.

Telot, J. (10 de 2012). *ResearchGate*. Obtenido de https://www.researchgate.net/figure/Figura-4-Diagrama-de-despliegue-UML_fig2_319234541

Testing, M. (6 de Agosto de 2018). *Mundo Testing*. Obtenido de https://mundotesting.com/etapas-de-pruebas-de-software-etapas-del-testing/

Torres, L. C. (16 de Febrero de 2012). Refactorización de Marcos Orientados a Objetos hacia Arquitecturas MVC. Mexico.

TutorialsPoint. (2016). *Ingeniería de Software tutorial*. Obtenido de Herramientas de Análisis y de Diseño: https://www.tutorialspoint.com/es/software_engineering/software_analysis_design_tools.htm

Ulloa, D. G. (Noviembre de 2014). ESTUDIO DE METODOLOGÍAS PARA ESTANDARIZAR EL DESARROLLO DE SOFTWARE EN EL DEPARTAMENTO DE INFORMÁTICA EN LA PASTORAL SOCIAL CARITAS DE LA DIÓCESIS DE AMBAT. Ambato, Ecuador.

Vivanco Villamar, A. A. (Agosto de 2011). Evaluación de calidad del sistema integrado para casas de valores SICAV de la bolsa de valoresde Quito utilizando la norma ISO/IEC 14598. Quito, Ecuador.

Wikipedia. (2019). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Pruebas_de_validaci%C3%B3n#targetText=Las%20pruebas%20de%20validaci%C3%B3n%20en,y%20que%20logra%20su%20cometido.

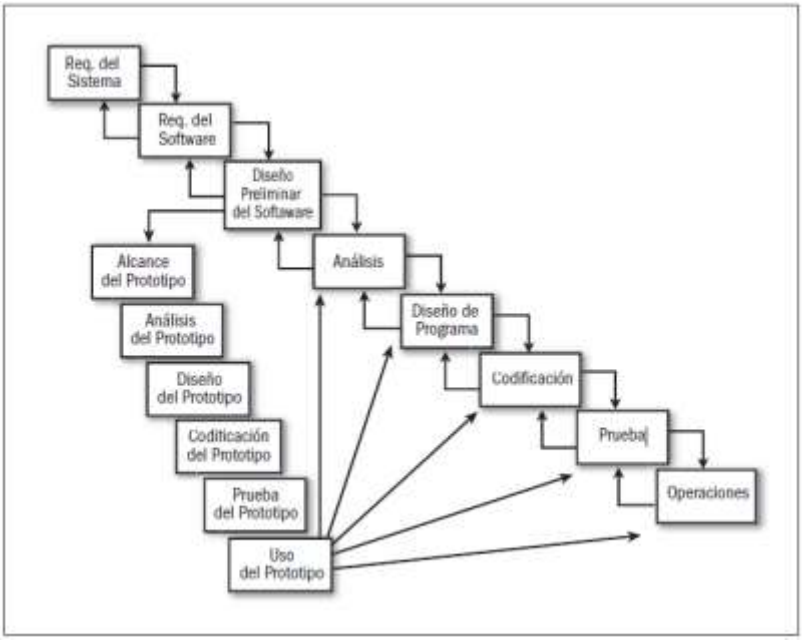
Yan Liu & Ian Gorton. (2005). Performance Prediction of J2EE Applications Using Messaging Protocols. *IEEE*, 16.

Zambrano, A. N. (junio de 2005). HERRAMIENTA PARA EL ANÁLISIS DE REQUERIMIENTOS DENTRO DE LA PEQUEÑA EMPRESA DESARROLLADORA DE SOFTWARE. *HERRAMIENTA PARA EL ANÁLISIS DE REQUERIMIENTOS*. Bogotá.

ANEXOS

Anexo N° 1: Modelos del Ciclo de Vida de Software

Ilustración 29: Modelo en Cascada Pura



Fuente: (Roberth, 2012)

Ilustración 30: Modelo en Cascada Pura

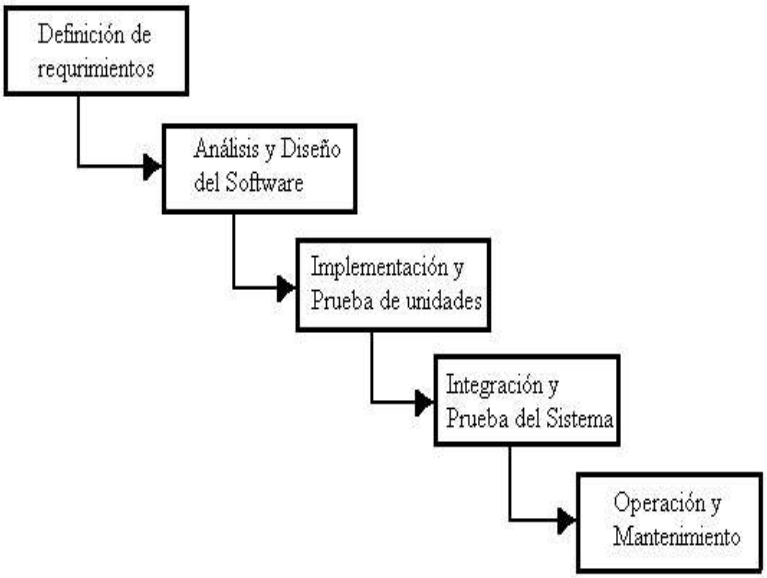
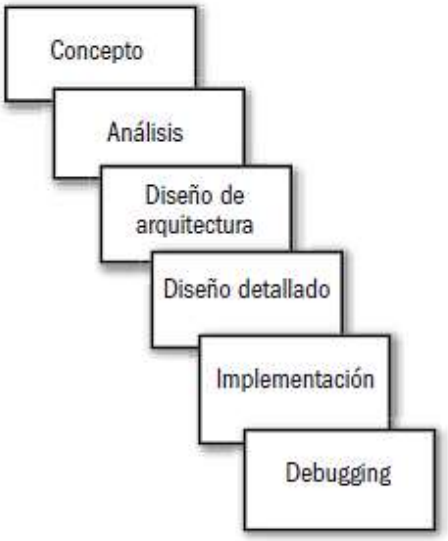
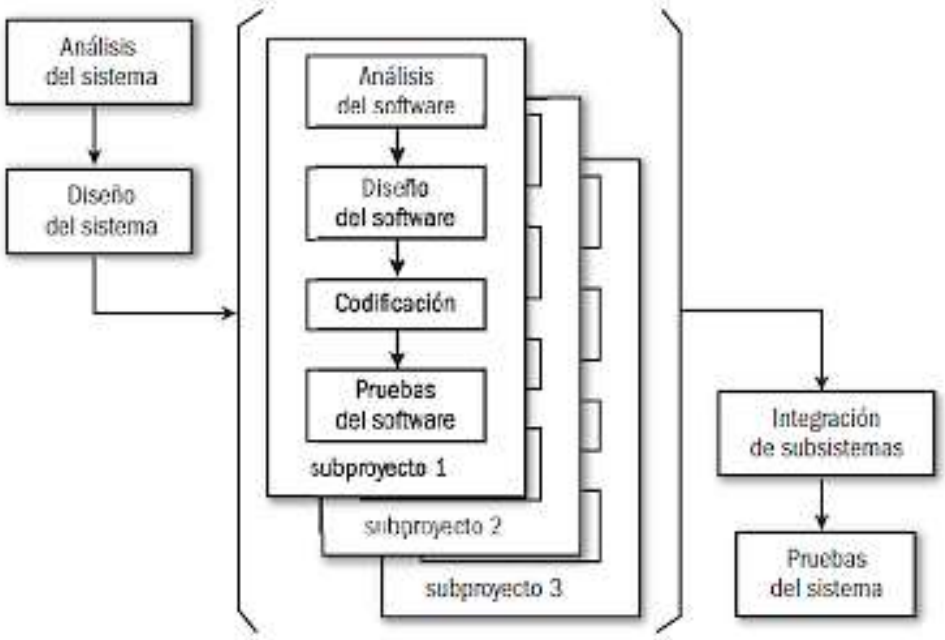


Ilustración 31: Modelo en Cascada con Fases Solapadas



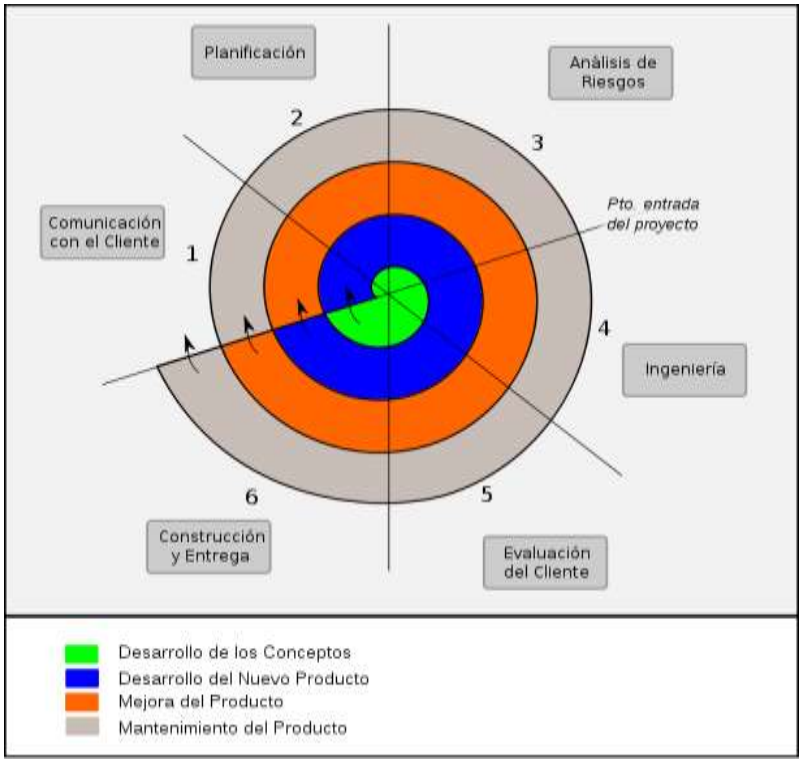
Fuente: (Roberth, 2012)

Ilustración 32: Modelo en Cascada con Subproyectos



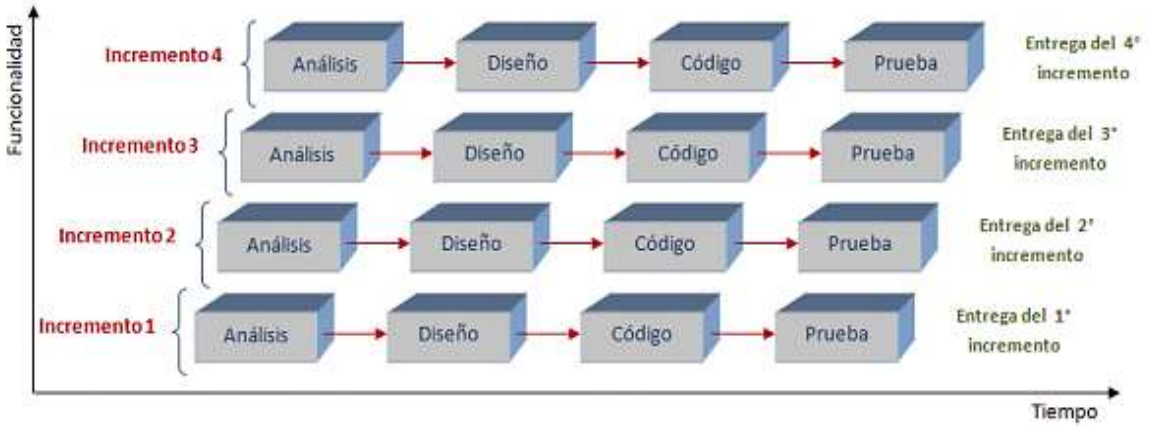
Fuente: (Roberth, 2012)

Ilustración 33: Modelo en Espiral



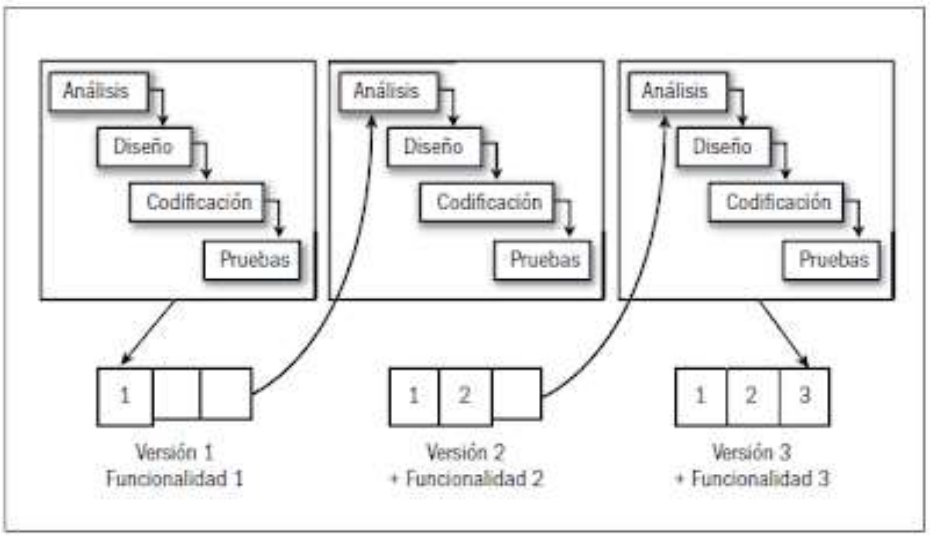
Fuente: (Grijalva, 2012)

Ilustración 34: Modelo Incremental



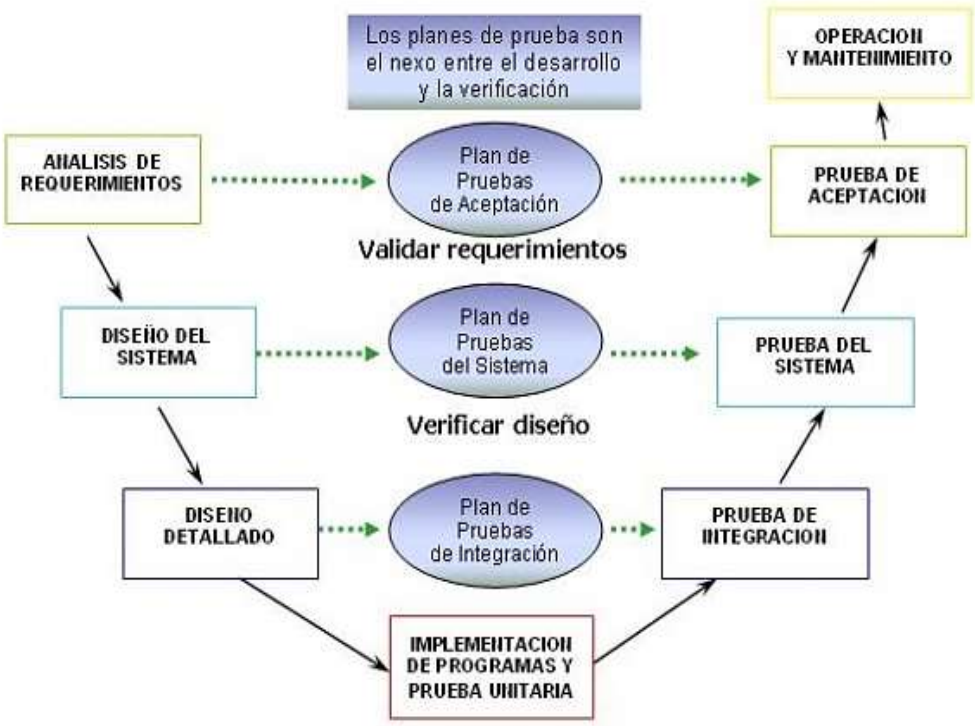
Fuente: (Marichelo, 2016)

Ilustración 35: Modelo evolutivo o iterativo



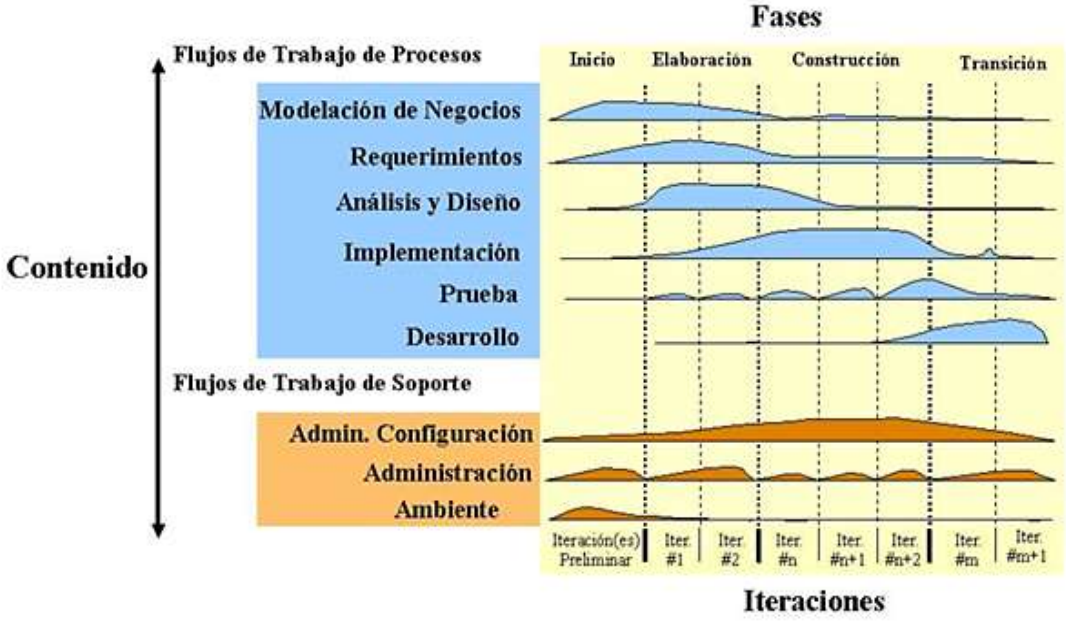
Fuente: (Roberth, 2012)

Ilustración 36: Modelo en Cascada V



Fuente: (Hernández, 2011)

Ilustración 37: Metodología RUP (Proceso Unificado Racional)



Fuente: (Francisco Apodaca, 2012)

Anexo N° 2: Métricas de calidad externa e interna (Anexo 24 – Obs.4 y Obs 5)

Ilustración 38: Métricas de calidad externa para Adecuación funcional (Funcionalidad)

Métricas para la característica de calidad Adecuación Funcional								
Subcaracterí sticas	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Propósito de la métrica de calidad**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de medida**	**Recurso s utilizado s**
Completitud funcional	Completitud de la implementaci ón funcional	Externa	¿Cuán completa es la implementación con respecto al uso del sistema?	Contar el número de las funciones referidas al sistema y el número de las funciones que faltan o están incorrectas.	X = A / B A = Número de funciones que están incorrectas o que no fueron implementadas. B = Número de las funciones instaladas en el sistema. Dónde: B > 0	0 ≤ X ≤ 1 El más cercano a 0 es el mejor.	X = Contable/ Contable A = Contable B = Contable	Observac ión

Fuente: (IEEE, Ingeniería de sistemas y software: Requisitos y evaluación de calidad de sistemas y software, 2011)

Ilustración 39*:* Métricas de la calidad externa para Facilidad de uso (Usabilidad)

Métricas para la característica de calidad de Facilidad de uso								
Subcaracterí stica	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Propósito de la métrica de calidad**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de medida**	**Recursos utilizados**
Capacidad de ser entendido	Efectividad de la documentaci ón del usuario o ayuda del sistema	Externa	¿Qué cantidad de funciones están descritas correctamente en la documentación del usuario?	Contar el número de funciones descritas correctamente y contar el número total de funciones implementadas	X = A / B A= N° de funciones descritas correctamente B = Número total de funciones implementadas Dónde: B > 0	0 ≤ X ≤ 1 El más cercano a 1 es el mejor	X = Contable/ Contable A = Contable B = Contable	Observaci ón
Operatividad	Claridad del mensaje	Externa	¿Qué cantidad de mensajes son auto explicativo?	Contar el número de mensajes implementados con explicaciones claras y el número total de mensajes implementados	X = A / B A = N° de mensajes Implementa os con explicaciones claras B = N° total de mensajes Implementa os Dónde: B > 0	0 ≤ X ≤ 1 El más cercano a 1 es el mejor	X = Contable/ Contable A = Contable B = Contable	Observaci ón
Estética de la interfaz del usuario	Personalizac ión de la apariencia de la interfaz del usuario.	Externa	¿Qué cantidad de los elementos de la interfaz de usuario pueden ser personalizados en apariencia?	Contar el número de tipos de elementos de interfaz que pueden ser personalizado y contar el número total de tipos de elementos de interfaz.	X = A/B A= N° de elementos de interfaz que pueden ser personalizados. B = N° total de elementos de interfaz Dónde: B > 0	0 ≤ X ≤ 1 El más cercano a 1 es el mejor	X = Contable/ Contable A = Contable B = Contable	Observaci ón

Ilustración 40: Métricas de calidad externa para Fiabilidad

Métricas para la característica de calidad de la Fiabilidad								
Subcaracterísti ca	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Propósito de la métrica de calidad**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de medida**	**Recurs os utilizad os**
Madurez	Eliminación de errores	Externa	¿Cuántos errores detectados han sido corregidos?	Contar el número de fallas corregidas en la fase de diseño/codificació n/pruebas y el número de fallas detectadas en las pruebas.	X = A / B A = Número de fallas corregidas en la fase de diseño/codificació n/prueba. B = Número de fallas detectadas en las pruebas. Dónde: B > 0	0 ≤ X ≤ 1 Cuanto más se acerque a 1 es el mejor.	X = Contable/ Contable A = Contable B = Contable	Observa ción
	Cobertura de pruebas	Externa	¿Cuántos casos de prueba requeridos han sido ejecutados durante la etapa de pruebas?	Contar el número de casos de pruebas realizados en un escenario de operación durante la prueba y el número de casos de prueba a ser realizados para cubrir los requerimientos	X = A/B A = Número de casos de pruebas realizados en un escenario de operación durante la prueba. B = Número de casos de prueba a ser realizados para cubrir los requerimientos. Donde B > 0	0 ≤ X ≤ 1 El más cercano a 1 es el mejor	X = Contable/ Contable A = Contable B = Contable	Observa ción
	Tiempo medio entre fallos.	Externa	¿Cuál es la frecuencia en que el sistema falla en la operación?	Tomar el tiempo de operación y contar el número total de fallas detectadas actualmente	X = A/T A = Número total de fallas detectadas actualmente T = Tiempo de operación Donde T > 0	X = A/T El más cercano a 0/t es el mejor.	X = Contable/ Tiempo A = Contable B = Tiempo	Observa ción.
Disponibilidad	Tiempo de servicio	Externa	¿Cuál es el tiempo de servicio del sistema que proporciona realmente?	Tomar el tiempo de servicio del sistema que se proporciona actualmente y tomar el tiempo de servicio del sistema regulado en el cronograma operacional.	X = A/B A = Tiempo de servicio del sistema que se proporciona actualmente B =. tiempo de servicio del sistema regulado en el cronograma operacional Donde B > 0	0 ≤ X ≤ 1 El más cercano a 1 es el mejor	X = Tiempo/ Tiempo A = Tiempo B = Tiempo	Observa ción
Tolerancia a fallos	Redundancia	Externa	¿Cuántos tipos de componentes del sistema son instalados de forma redundante para evitar fallo en el sistema?	Contar el número total de tipos de componentes y el número de tipos de componentes instalados de forma redundante.	X = A/B A = Número componentes/sist emas instalados de forma redundante. B =. Número total de componentes/sist emas instalados. Donde B > 0	0 ≤ X ≤ 1 El más cercano a 1 es el mejor	X = Contable/ Contable A = Contable B = Contable	Observa ción

Fuente: (IEEE, Ingeniería de sistemas y software: Requisitos y evaluación de calidad de sistemas y software, 2011)

Ilustración 41: Métricas de calidad externa para Eficiencia en el desempeño (Rendimiento)

Métricas para la característica de calidad de la Eficiencia en el desempeño								
Subcaracterísti ca	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Propósito de la métrica de calidad**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de medida**	**Recursos utilizado s**
Comportamient o del tiempo	Tiempo de respuesta	Externa	¿Cuál es el tiempo estimado para completar una tarea?	Tomar el tiempo desde que se envía la petición hasta obtener la respuesta.	X = B – A A = Tiempo de envío de petición. B = Tiempo en recibir la primera respuesta.	0 ≤ X ≤ 1 El más cercano a 0 es el mejor. Donde el peor caso es ≥ 15t.	X = Tiempo/ Tiempo A = Tiempo B = Tiempo	Observaci ón
	Tiempo de espera	Externa	¿Cuál es el tiempo desde que se envía una instrucción, para que inicie un trabajo hasta que lo completa?	Tomar el tiempo cuando se inicia un trabajo y el tiempo en completar el trabajo.	X = B – A A = Tiempo cuando se inicia un trabajo. B = Tiempo en completar un trabajo.	0 ≤ X ≤ 1 El más cercano a 0 es el mejor. Donde el peor caso es ≥ 15t.	X = Tiempo/ Tiempo A = Tiempo B = Tiempo	Observaci ón
Utilización de recursos	Utilización de CPU	Externa	¿Cuánto tiempo el CPU es usado para realizar una tarea dada?	Tomar el tiempo de operación y la cantidad de tiempo de CPU que se usa para realizar una tarea.	X = B - A A = La cantidad de tiempo de CPU que realmente es usado para realizar una tarea. B =. tiempo de operación Donde B > 0	0 ≤ X ≤ 1 Cuanto más se acerque a 0 es mejor. Donde el peor caso es ≥ 15t.	X = Tiempo / Tiempo A = Tiempo B = Tiempo	Observaci ón

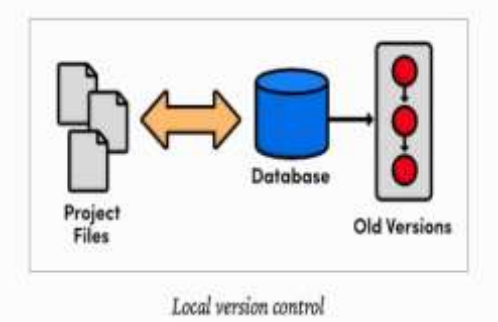
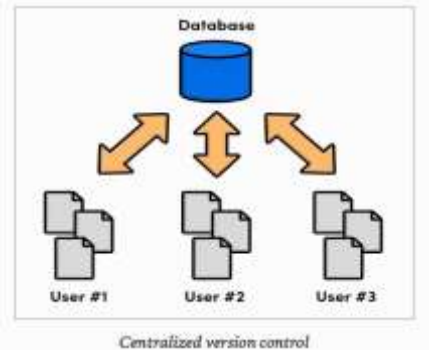
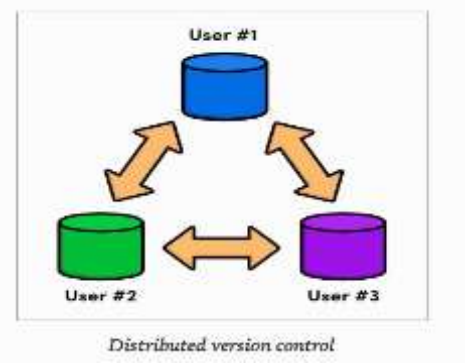
Fuente: (IEEE, Ingeniería de sistemas y software: Requisitos y evaluación de calidad de sistemas y software, 2011)

Ilustración 42: Métricas de calidad externa para Mantenibilidad

Métricas para la característica de calidad Mantenibilidad								
Subcaracteri stica	**Métrica**	**Fase del ciclo de vida de calidad del producto**	**Propósito de la métrica de calidad**	**Método de aplicación**	**Fórmula**	**Valor deseado**	**Tipo de medida**	**Recursos utilizados**
Capacidad de ser modificado	Complejida d de modificació n	Externa	¿Con qué facilidad el desarrollador puede modificar el software para resolver problemas?	Tomar el tiempo de trabajo que le toma al desarrollador modificar y contar el número de modificaciones	X = A/T A = Nº de modificaciones. T = Tiempo de trabajo que le toma al desarrollador modificar. Dónde; T > 0	X = A/T El más lejano a 0/t es el mejor.	X = Contable/ Tiempo A = Contable T = Tiempo	Observaci ón

Fuente: (IEEE, Ingeniería de sistemas y software: Requisitos y evaluación de calidad de sistemas y software, 2011)

Anexo N° 3: Cuadro comparativo de tipos de sistemas de control de versiones

Sistema de Control de Versiones Local	Sistemas de Control de Versiones Centralizados	Sistemas de Control de Versiones Distribuidos
		
Puedes copiar archivos a otros directorios, especificando variaciones en los nombres (archivo1, archivo2, archivo-final… etc.)	tienen un único servidor que contiene todos los archivos versionados, y varios clientes descargan los archivos desde ese lugar central	Es aquí donde entran los sistemas de control de versiones distribuidos, los clientes no sólo descargan la última instantánea de los archivos: también replican completamente el repositorio.
Es fácil olvidar en qué directorio te encuentras, y guardar accidentalmente en el archivo equivocado o sobrescribir archivos que no querías.	Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han llevado copias de seguridad adecuadamente, pierdes absolutamente todo.	Así, si un servidor muere, y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo.
cuando tienes toda la historia del proyecto en un único lugar, te arriesgas a perderlo todo.	cuando tienes toda la historia del proyecto en un único lugar, te arriesgas a perderlo todo.	cada miembro del equipo tiene una copia local del sistema de control de versiones.
Cada vez que tienes que realizar una modificación está propensa a errores.	Cada vez que tienes que realizar una modificación tienes que mirar en la nube la última versión, descargarla, hacer las modificaciones, volverla a subir, y si tu conexión a internet es lenta, tu trabajo será muy engorroso.	Cada vez que realizamos una modificación en nuestra copia ésta se reporta a la copia principal, donde es gestionada convenientemente por el administrador del proyecto.
	Ejemplos: CVS, Subversión, Perforce	como Git (el más usado), Mercurial, Bazaar o Darcs,PlasticSCM

Anexo N° 4: Formato "Solicitud de requerimientos de software"

<table>
<tr><td rowspan="2"> 
Municipalidad Provincial de Chiclayo</td><td colspan="3">FORMATO "SOLICITUD DE REQUERIMIENTO DE SOFTWARE"</td></tr>
<tr><td>Código: SR</td><td>Versión: 1.0</td><td>Año:</td></tr>
<tr><td colspan="4">SOLICITUD DE REQUERIMIENTO</td></tr>
<tr><td>Número de Solicitud:</td><td colspan="3"></td></tr>
<tr><td>Fecha de Solicitud:</td><td colspan="3"></td></tr>
<tr><td>Tipo de Requerimiento:</td><td colspan="3"></td></tr>
<tr><td>Título del Requerimiento:</td><td colspan="3"></td></tr>
<tr><td>Unidad Orgánica:</td><td colspan="3"></td></tr>
<tr><td>Usuario Solicitante:</td><td colspan="3"></td></tr>
<tr><td>E-mail del solicitante:</td><td colspan="3"></td></tr>
<tr><td colspan="4">Descripción del requerimiento</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td colspan="4">Origen o justificación de la necesidad</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td colspan="4">Criticidad (Grado de importancia)</td></tr>
<tr><td>Alta ☐</td><td colspan="2">Media ☐</td><td>Baja ☐</td></tr>
<tr><td colspan="4">Áreas Involucradas</td></tr>
<tr><td colspan="2">_____
Unidad Orgánica</td><td colspan="2">_____
Dirección General</td></tr>
</table>

Anexo N° 5: Formato "Entrevista"

<table>
<tr><td rowspan="2"> 
Municipalidad Provincial de Chiclayo</td><td colspan="3">FORMATO "ENTREVISTA"</td></tr>
<tr><td>Código: FE</td><td>Versión:</td><td>Año:</td></tr>
<tr><td colspan="4">ENTREVISTA</td></tr>
<tr><td>Fecha:</td><td></td><td>Hora:</td><td></td></tr>
<tr><td>Resumen:</td><td colspan="3"></td></tr>
<tr><td colspan="4">PREGUNTAS:</td></tr>
<tr><td colspan="4">1. ¿Cuál son las actividades principales qué realizan en su área?
2. ¿Cuántas personas son en su área?
3. ¿Qué tipo de información manejan(especifique)?
4. ¿Cómo manejan la información?
5. ¿Quiénes tienen acceso a la información?
6. ¿Qué modalidades utilizan para archivar la información?
7. ¿Cada cuánto actualizan los registros de la información?
8. ¿Cuál es la problemática que buscan solucionar a través del sistema (o modulo o requerimiento) que
solicitan?
9. ¿Cuáles son los pasos para el proceso que realizaran y que resultados quieren obtener?
10. ¿Quiénes tendrán acceso al sistema de información?
11. ¿Han utilizado anteriormente algún sistema parecido?
12. ¿Necesitarán de alguna información de la empresa en particular ya sea impresa o en línea?
13. ¿Cuáles son sus expectativas con respecto a la facilidad de uso de este sistema?
14. ¿La visualización o aspecto del sistema como le gustaría q fuera? o talvez ¿le gustaría como
algún sistema que usa o ha utilizado?
15. ¿Qué le ha dificultado más cuando ha interactuado con un sistema?
(ejemplos: tiempos de respuesta, no ubicaba los botones, la letra muy pequeña, no entendía los datos que pedía, etc.)</td></tr>
</table>

Anexo N° 6: Formato "Identificación de necesidades"

	FORMATO "IDENTIFICACIÓN DE NECESIDADES"		
Municipalidad Provincial de Chiclayo	**Código:** IN	**Versión:**	**Año:**
IDENTIFICACIÓN DE NECESIDADES			
Fecha:		**Hora:**	
Resumen:			

Identificación de necesidades	**Nivel de prioridad**		
	Primario	**Secundario**	**Opcional**
✓			
✓			
✓			
✓			
✓			
✓			
✓			
✓			
✓			
✓			
Justificación o razones:			

CATEGORÍA	**DESCRIPCION**	**EXPLICACION**
Primario (P)	No debe faltar	Las necesidades son primordiales
Secundario (S)	Es útil, pero no indispensable	Las necesidades secundarias sirven de complemento a las necesidades primordiales
Opcional (O)	Es alternativo, novedoso, pero no necesario	Las necesidades son opcionales

Anexo N° 7: Formato "Definición y análisis de requerimientos de usuario"

Municipalidad Provincial de Chiclayo	**FORMATO "DEFINICIÓN Y ANÁLISIS DE REQUERIMIENTOS DE USUARIO"**		
	Código: RQU	**Versión:**	**Año:**

DEFINICIÓN Y ANÁLISIS DE REQUERIMIENTOS DE USUARIO

Listado de Requerimientos de usuario				**Analizar requerimiento**	**Analizar objetivo estratégico**
ID RQ	**TIPO DE RQ**	**REQUERIMIENTO**	**DESCRIPCIÓN**	**CARACTERÍSTICA INDIVIDUAL**	**DESCRIPCIÓN DE OBJETIVO**
Identificador del Requerimiento	Funcional / no funcional	Título del requerimiento	Breve descripción del requerimiento	Revisar la tabla de características individuales	Describa que objetivo institucional o del negocio desea apoyar con ese requerimiento

Restricciones:

Las restricciones pueden resultar de instancias o áreas de soluciones definidas por los interesados o de decisiones de implementación tomadas en niveles más altos de la estructura jerárquica del sistema.

Deficiencias o Debilidades:

Problemas encontrados en los requerimientos de usuario

Anexo N° 8: Formato "Acta de Reunión"

Municipalidad Provincial de Chiclayo	**Formato "Acta de Reunión"**				
	Código: AC	**Versión**	**Año:**		
ACTA DE REUNIÓN					
Nombre del proyecto:					
Fecha:		**Hora de Inicio:**		**Hora de cierre:**	
Lugar:					
Listado de asistentes:					

Nombres y Apellidos	*Área proveniente*

Temas:

N°	*Tema*	*Responsable*	*Duración*

Acuerdos:

Ítem	*Acuerdos*	*Responsable*	*Fecha de entrega*

Firmas de Participantes

Participantes	*Firma*

Anexo N° 9: Formato "Registro detallado por requerimiento del sistema"

<table>
<tr><td rowspan="2"> 
Municipalidad Provincial de Chiclayo</td><td colspan="4">FORMATO "REGISTRO DETALLADO POR REQUERIMIENTO DEL SISTEMA"</td></tr>
<tr><td>Código: RRQ</td><td>Versión:</td><td colspan="2">Año</td></tr>
<tr><td colspan="6">REGISTRO DETALLADO POR REQUERIMIENTO DEL SISTEMA</td></tr>
<tr><td>Versión:</td><td>Versión del requerimiento</td><td>Solicitante:</td><td colspan="3">Especifique el área, entidad o cargo solicitante del requerimiento</td></tr>
<tr><td>Identificador del requerimiento:</td><td>abreviatura que identifica el requerimiento</td><td>Nombre del requerimiento:</td><td colspan="3">Nombre corto del requerimiento a través de una denominación clara</td></tr>
<tr><td rowspan="5">Tipo de requerimiento</td><td rowspan="5">Defina el tipo de requerimiento funcional, no funcional (de calidad/de interfaz externa)</td><td rowspan="5">Estado:</td><td>Estado</td><td colspan="2">Fecha</td></tr>
<tr><td>Especificado</td><td colspan="2">Incluir la fecha en que se detalló el requerimiento.</td></tr>
<tr><td>Diseñado</td><td colspan="2">Incluir la fecha en que se diseñó el requerimiento.</td></tr>
<tr><td>Implementado</td><td colspan="2">Incluir la fecha en que el requerimiento fue implementado</td></tr>
<tr><td>Aprobado</td><td colspan="2">Incluir la fecha en que el requerimiento fue aprobado.</td></tr>
<tr><td colspan="6">Restricciones de Implementación</td></tr>
<tr><td colspan="6">Estas restricciones incluyen las condiciones bajo las cuales el sistema debe ser capaz de realizar la función.</td></tr>
<tr><td colspan="6">Comentarios</td></tr>
<tr><td colspan="6"></td></tr>
</table>

Anexo N° 10: Formato "Definición y análisis de requerimientos del sistema"

Municipalidad Provincial de Chiclayo	**FORMATO "DEFINICIÓN Y ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA"**								
	Código: ARQ	**Versión:**			**Año:**				
DEFINICIÓN Y ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA									
Lista de Requerimientos	**Característica individual del requerimiento** **Marcar con un aspa (X)**								
	N	**IG**	**I**	**CO**	**COM**	**S**	**F**	**R**	**V**
Requerimientos Funcionales									
Requerimientos No Funcionales									
RESUMEN DE REQUERIMIENTOS	**Característica grupal de los requerimientos**								
	COMPLETO	**CONSISTENTE**	**ASEQUIBLE**	**ENCERRADO**					
Deficiencias o Debilidades encontradas									

Característica individual del requerimiento

(base para el cuadro de arriba)

Característica	Abrev.	Definición
Necesario	(N)	El requisito es actualmente aplicable y no ha quedado obsoleto por el paso del tiempo.
Implementación gratuita	(IG)	El requerimiento, al abordar lo que es necesario y suficiente en el sistema, evita colocar restricciones innecesarias en el diseño arquitectónico.
Inequívoco	(I)	El requisito se establece de tal manera que se puede interpretar de una sola manera. El requisito se establece de manera simple y es fácil de entender.
Consistente	CO)	El requisito está libre de conflictos con otros requisitos.
Completo	(COM)	El requisito establecido no necesita más amplificación porque es medible y describe suficientemente la capacidad y las características para satisfacer las necesidades de las partes interesadas.
Singular	(S)	La declaración de requisito incluye solo un requisito sin uso de conjunciones.
Factible	(F)	El requisito es técnicamente alcanzable, no requiere avances tecnológicos importantes y se ajusta a las limitaciones del sistema (por ejemplo, costo, cronograma, técnico, legal, reglamentario) con un riesgo aceptable.
Rastreable	(R)	El requisito es rastreable hacia las declaraciones de las partes interesadas documentadas específicas de la necesidad, y también se puede rastrear hacia los requisitos específicos en la especificación de requisitos.
Verificable	(V)	El requisito tiene los medios para demostrar que el sistema satisface el requisito especificado.

Anexo N° 11: Formato "Especificación de requerimientos"

Municipalidad Provincial de Chiclayo	FORMATO "ESPECIFICACIÓN DE REQUERIMIENTOS"		
	Código:HRQ	**Versión**	**Año**

ESPECIFICACIÓN DE REQUERIMIENTOS

Historial de Modificacion

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
N°	Motivo de modificación	Nombre y Apellido	dd/mm/aaaa

Entorno Tecnológico Actual

- **Descripción del Entorno de Hardware Actual**
  Describe servidores, estaciones de trabajo, redes, etc., que pueda tener impacto sobre el sistema software a desarrollar.
- **Descripción del Entorno de Software Actual**
  Describe sistemas operativos, sistemas de gestión de bases de datos, servidores de aplicaciones, etc.

Descripción de los subsistemas o módulos del sistema a desarrollar

Nombre:	nombre descriptivo
Id:	Identificador único del módulo
Versión:	N° de versión
Dependencias:	De que subsistema o modulo depende
Descripción:	Descripción del subsistema
Importancia:	importancia para el usuario
Comentarios	comentarios adicionales sobre el subsistema

Catálogo de requerimientos

Casos de uso del sistema

Definir actores del sistema:

Debe contener las especificaciones de los actores que se hayan identificado en los casos de uso, es decir, los diferentes tipos de usuarios y otros sistemas con los que deba interactuar el sistema a desarrollar.

Especificaciones de casos de uso:

Debe contener las especificaciones de los casos de uso del sistema que se hayan identificado.

Especificación de Caso de uso		
Identificador del caso de uso:		<CUn° ejem: CU1>
Versión:		<asígnese una numerología para la versión ejem: 1.0 (dd/mm/aaaa)>
Nombre del caso de uso:		<nombre claro y especifico del requerimiento>
Descripción del caso de uso:		<Describa la función del requerimiento>
Tipo:		<Necesario/ Deseable>
Actores:		<Defina quienes serán los actores
Dependencias:		<Defina de que requerimientos depende> 1. 2.
Precondiciones:		<defina si existe alguna condición antes de iniciar la función, ejem: el usuario debe tener una cuenta>
Secuencia (flujo de eventos):		
N°	Acción del actor	Respuesta del Sistema
1 2	<acción realizada por el actor>	<acción realizada por el sistema>
Postcondición:	< mensaje de éxito>	
N°	Excepción de la acción del actor	Excepción de la respuesta del sistema
1 2	la acción no cumple con alguna condición	mensaje que emite el sistema

Criterios de Aceptación		
Defina en que criterio el requerimiento se ha llevado a cabo correctamente		
Comentarios		

Resumen de Requerimientos

Identificacion	Tipo de Requerimiento	Nombre	Descripción	Prioridad
Identificador unico del requerimiento	Funcional/no Funcional	Nombre corto del requerimiento	Descripcion breve del requerimiento	Intensidad de importancia del requerimiento

Validación y Verificación

a) ***Planeación:***

Agendar Reunión			
Fecha	Lugar	Material	Tiempo requerido

b) Preparación:

Previo a la reunión, entregar el material a los participantes para su revisión

c) Reunión de Inspección:

- Leer cada requerimiento,
- Solicitar a los participantes que indiquen los problemas y errores encontrados y
- Registrarlos.

d) ***Corrección:*** **Resolver problemas o errores**

Id de Requerimiento	Nombre corto de Requerimiento	Problema o Error Encontrado	Solución:
Identificador del requerimiento	Nombre específico del requerimiento	Describa el error encontrado	Eliminar o modificar requerimientos

e) Verificación:

Código	Aprobado por	Fecha de Aprobación
<<Código del Requerimiento>>	<<Nombre del usuario que aprobó el requerimiento>>	<<Fecha de Aprobación del requerimiento>>

Matriz de Trazabilidad

Requerimiento funcional	**Elemento de diseño**	**Elemento de código**	**Caso de prueba**	**Petición de cambio**
<<Descripción del requerimiento funcional>>	<<Descripción de os elementos de diseño relacionados con el requerimiento>>	<<Descripción de os elementos de código relacionados con el requerimiento>>	<<Caso de prueba del requerimiento funcional>>	<<Referencia a las peticiones de cambio relacionadas con el requerimiento (en caso de existir)>>

Anexo N° 12: Formato "Acta de conformidad de usuario"

Municipalidad Provincial de Chiclayo	**FORMATO "ACTA DE CONFORMIDAD DE USUARIO"**		
	Código: ACU	**Versión:**	**Año**

ACTA DE CONFORMIDAD DE USUARIO

Fecha:		**Hora:**	
Área Solicitante:			
Analista de Sistemas:			

En la presente acta de conformidad, el usuario debe constatar que los requerimientos solicitados han sido plasmados correctamente en la reunión, así como también plasmar sus observaciones en caso las tuviera.

Además, si en el transcurso de los días se presentase nuevos requerimientos o posibles cambios, el usuario se compromete a comunicárselo a la brevedad al área de desarrollo de sistemas para evitar inconsistencias y aplazamientos en la entrega de lo solicitado.

Conformidad:	
Se da Conformidad ☐	Se presentan observaciones ☐

Observaciones:

Firmas:	
_____ **Jefe de Sistemas**	_____ **Dirección de Área Solicitante**

Anexo N° 13: Formato "Petición de cambio"

Municipalidad Provincial de Chiclayo	FORMATO "PETICIÓN DE CAMBIO"		
	CÓDIGO: PC	**Versión:1.0**	**Año**
PETICIÓN DE CAMBIO			
Fecha:		**Identificador:**	
Estado:			
DATOS DEL PROYECTO			
Nombre proyecto:		**Módulo:**	
DATOS DEL SOLICITANTE			
Nombre:		**Cargo:**	
Área:		**Correo:**	
REGISTRO DE LA PETICION DE CAMBIO			
Título:	<<Resumen de una línea de la petición de cambio>>		
Tipo de cambio:	<<Nuevo requerimiento/Cambio de requerimiento/eliminación de requerimiento>>		
Descripción de la petición			
Justificación:			
Prioridad:	<<Alta/media/baja>>		
ANÁLISIS Y EVALUACIÓN DE LA PETICIÓN DE CAMBIO			
Responsable:			
Descripción del análisis:			
Decisión adoptada:		**Fecha de decisión:**	
Responsable de decisión:		**Razón de decisión:**	
IMPLMENTACIÓN DE LA PETICIÓN DE CAMBIO			
Responsable:			
Fecha de inicio:		**Fecha fin**:	
Artefactos:			
CIERRE			
Cerrado por:		**Fecha:**	

Anexo N° 14: Formato "Arquitectura tecnológica del sistema"

Municipalidad Provincial de Chiclayo	FORMATO "ARQUITECTURA TECNOLÓGICA DEL SISTEMA"		
	Código: ATS	**Versión:1.0**	**Año:**
ARQUITECTURA TECNOLÓGICA DEL SISTEMA			
DESCRIPCION DEL ENTORNO TECNOLÓGICO			
Hardware: Describir procesadores, unidades de almacenamiento, estaciones de trabajo, etc. Software: Definir sistemas operativos, subsistemas, gestores de base de datos, etc. Comunicaciones: Definir líneas, capacidad de elementos de red, protocolos, etc.			
RESTRICCIONES TÉCNICAS			
Describir las restricciones técnicas derivadas de la tecnología seleccionada que afectan al diseño y construcción del sistema.			
DESCRIPCIÓN BREVE DE LA ARQUITECTURA DEL SISTEMA			
Describir los diferentes niveles de arquitectura utilizados, las tecnologías empleadas en cada uno de ellas, la comunicación entre los componentes desarrollados y la ubicación de los elementos en capas. En caso de utilizarse un modelo de arquitectura para representarla y documentarla, definir cual se usará.			
DISTRIBUCIÓN DEL SISTEMA			
Describir los diferentes niveles de arquitectura, mediante la definición de las principales particiones físicas del sistema, representados en nodos (partes del sistema) y comunicación entre nodos (interfaces entre las partes).			
DIAGRAMA DE LA ARQUITECTURA TECNOLÓGICA (VISTA FISICA)			
Hacer el diagrama de despliegue de la vista física de arquitectura			
PATRON DE DISEÑO			
Definir los estándares técnicos, nomenclatura, normas y recomendaciones que puedan condicionar el patrón de diseño del sistema.			

Anexo N° 15: Formato "Diseño del sistema"

Municipalidad Provincial de Chiclayo	FORMATO "DISEÑO DEL SISTEMA"		
	Código: DS	**Versión: 1.0**	**Año:**

DISEÑO DEL SISTEMA

Diseño Arquitectónico

Describir el diseño de acuerdo al patron que se utilizara para la arquitecutra

Diseño de Interfaz

- Estructura de menú(navegación)
- Diagrama de casos de uso, de actividad, de secuencia, de estado, de colaboración
- Prototipos de Pantalla

Diseño de componentes

- Diagrama de componentes

Diseño de Base de Datos

- Modelo conceptual (modelo entidad -Relación)
- Diagrama de clases
- Descripción de Tablas

Nombre de la Tabla:						
campo	tipo	null	Llave primaria	Valor por defecto	Llave foránea	Descripción
<<Nombre del atributo >>	<<Tipo de variable>>	SI/NO	SI/NO	<<código por defecto(opcional)>>	SI/NO	<<Describir lo que almacena el campo>>

- **Diccionario de datos**

✓ ***Diccionario de clases***

Nombre	Tipo	Descripción
<<Nombre de la clase>>	<<Pública>>	<<Funcionalidades que puede cumplir la clase>>

✓ ***Diccionario de atributos***

Nombre	Tipo	visibilidad	Descripción
<<Nombre del atributo de la BD>>	<<void/integer/array/string, etc.>>	<<Público>>	<<Para qué sirve el atributo>>

- ✓ ***Diccionario métodos de clase***

Nombre	Tipo de retorno	Parámetros	Descripción
<<Nombre del Método>>	<<void/integer/array/string, etc.>>	<<Parámetros que usa el método>>	<<Describe de que se encarga el método>>

MATRIZ DE TRAZABILIDAD

INTERFAZ	Nombre de la interfaz	
CASOS DE USO RELACIONADOS CON LA INTERFAZ	IDENTIFICADOR DEL CASO DE USO	NOMBRE DEL CASO DE USO

Casos de uso	Diagrama de Actividad	Diagrama de Secuencia	Diagrama de Colaboración	Diagrama de Estado
CURF1	<<marcar con un aspa si se realizó el diagrama>>	<<marcar con un aspa si se realizó el diagrama>>	<<marcar con un aspa si se realizó el diagrama>>	<<marcar con un aspa si se realizó el diagrama>>

Evaluación de la arquitectura

Existen 2 tipos de evaluaciones las cualitativas (ejemplo: cuestionarios, cheklists, escenarios (con casos de uso)) y cuantitativas (modelos matemáticos, métricas)

Para la evaluación cualitativa se puede utilizar esta plantilla:

<Nombre caso prueba>	**<ID del CP>**
ID Caso de Uso	
Autor del Caso de Prueba	*<Persona que elabora el caso de prueba>*

Nombre del Probador	*<Persona que ejecuta la prueba>*		
Fecha de Creación:		Fecha de Ejecución:	
Versión:			
Descripción			
<Descripción del caso de prueba>			
Prerrequisitos			
<Enumerar los prerrequisitos para la prueba>			
Pasos:			
<Pasos generales para la prueba, basados en los escenarios de los casos de uso, si existen.>			
Resultado esperado:			
<Resultado esperado de la prueba>			
Resultado obtenido:			
<Resultado obtenido de la ejecución del caso de prueba>			
Decisión de Aprobación del Caso de Prueba			
Aprobó ☐		Falló ☐	
Firma del Probador		Firma del Área Usuaria	
Fecha de Aprobación del Caso de Prueba:			

Para la evaluación cuantitativa se puede utilizar la siguiente plantilla

Nota: Registre los datos de acuerdo al atributo de calidad que se desea medir para ello revisar el Anexo 20:

TABLA DE MÉTRICAS	
Ítem	**Descripción**
SubCaracterística	Sub característica de calidad
Nombre de la métrica	Nombre asignado a la métrica de calidad.
Fase del Ciclo de vida del producto.	Fase del ciclo de vida: calidad interna, calidad externa y calidad en uso
Propósito de la métrica de calidad.	Motivo por el cual se selecciona la métrica.
Método de aplicación	Manera de cómo se va a aplicar la métrica
Formula y cálculo de datos	Establece la fórmula de medición y especifica los significados de los datos que se van a utilizar
Valor deseado	Proporciona el rango y los valores preferibles y recomendados.
Tipo de medida	Especifica en tipo de medida que se va seleccionar, como: tamaño (tamaño de la función, tamaño de la fuente), tiempo (lapso de tiempo, tiempo de usuario), contar (número de cambios, números de fallas).
Recursos utilizados	Específica los recursos que se utilizarán para poder medir cada métrica, entre los recursos utilizados, pueden estar: entrevistas a usuarios, código fuente, documentación, entre otras.

OBSERVACIONES

Anexo N° 16: Formato "Análisis del sistema"

Municipalidad Provincial de Chiclayo	**FORMATO "ANALISIS DEL SISTEMA"**		
	Código: AS	**Versión:**	**Año:**
ANÁLISIS DEL SISTEMA			
Fecha:		**Hora:**	
Responsable del análisis:			
Objetivos:			
Especifique los objetivos generales para la realización del análisis del sistema.			
Analizar:			
Revisar y analizar todos los documentos concernientes a la elaboración del sistema desde la obtención de requerimientos.			
Glosario:			
Crear un glosario con los términos, acrónimos o abreviaturas que usted crea convenientes para hacer más fácil el entendimiento de este documento.			
Observaciones:			
Detalle las observaciones encontradas en el análisis.			
Conclusiones:			
Detalle las conclusiones una vez realizado el análisis del sistema.			

Anexo N° 17: Formato "Mantenimiento del sistema"

Municipalidad Provincial de Chiclayo	**FORMATO "MANTENIMIENTO DEL SISTEMA"**		
	Código: MS	**Versión:**	**Año:**

MANTENIMIENTO DEL SISTEMA

Información General del proyecto de Software			
Nombre del Proyecto:			
Fecha:			
Solicitante:			
Cargo:			
Descripción del requerimiento			
Descripción detalla del requerimiento, documentada por el usuario.			
Análisis del Requerimiento			
Factible			
No Factible		¿Por qué?	
Tipo de Mantenimiento			
Correctivo ( )	**Adaptativo ( )**	**Perfectivo ( )**	**Preventivo ( )**
Control de Cambios (En caso aplique)			
Nombre del sistema:			
Descripción del trabajo realizado			
Descripción del trabajo realizado por el Analista de Sistemas			
Fecha de inicio:		**Fecha de término:**	
Constancia de Mantenimiento			

Jefe de Sistemas	Responsable del Área Solicitante	Analista de Sistemas

Anexo N° 18: Formato "Incidentes y reclamos"

Municipalidad Provincial de Chiclayo	**FORMATO "INCIDENTES Y RECLAMOS"**		
	Código: IR	**Versión:**	**Año:**
INCIDENTES Y RECLAMOS			
Fecha de Registro:	Fecha en que se registró el incidente o reclamo		
Área:	Área usuaria que presento el incidente o reclamo		
Descripción del incidente o reclamo			
Detalle específicamente cual es el incidente o reclamo presentado.			
Origen:	Que originó el incidente o reclamo		
Clasificar el incidente:	Es decir, si el incidente o reclamo se dio por un error en el código, error funcional, o error en diseño		
Estado:	En qué estado se encuentra, es decir si aún está pendiente o ya se le ha dado solución		
Fecha comprometida:	Fecha en la que se dará solución al incidente o reclamo.		
# veces:	Cantidad de veces que se presenta un incidente con las mismas características.		
Responsable	Persona quien dará solución al incidente		
Conclusiones:			
Detalle las conclusiones una vez realizado el análisis del sistema.			

Anexo N° 19: Formato "Conformidad de capacitación"

Municipalidad Provincial de Chiclayo	FORMATO DE CONFORMIDAD DE CAPACITACIÓN		
	Código: FCC	**Versión 1.0**	**Año:**

CONFORMIDAD DE CAPACITACIÓN

Fecha de Capacitación:			
Duración:			
Capacitador:			
Área usuaria:			
Persona encargada de la Conformidad			
Apellidos y Nombres:			
Cargo:		**Correo Electrónico:**	

Capacitación Realizada

Básico ☐ **Medio** ☐ **Avanzado** ☐

Tema

Preguntas

1. ¿Está conforme con la capacitación que se dio?

SI ☐ **NO** ☐

Si su respuesta es NO, por favor indicar los motivos.

_____
_____

2. ¿El material utilizado en la capacitación fue dinámico, claro, conciso?

Poco ( ) Regular ( ) Mucho ( ) Bastante ( )

3. ¿El capacitador logró despejar sus dudas?

SI ☐ **NO** ☐

Si su respuesta es NO, por favor indicar los motivos.

_____
_____

4. Se ha capacitado a ( ), personas de su Entidad, ¿está conforme? **Si ( ) , No ( )**

Si su respuesta es NO, por favor indicar los motivos.

_____
_____

Áreas Involucradas

_____ **Participante**	_____ **Capacitador**

Anexo N° 20: Formato "Plan de pruebas"

Municipalidad Provincial de Chiclayo	**FORMATO "PLAN DE PRUEBAS"**		
	Código: PP	**Versión:**	**Año:**
PLAN DE PRUEBAS			
Información general			
Nombre del Proyecto:			
Fecha:			
Tipo de Prueba:	Unitarias ( ) Integración ( ) Verificación ( ) Validación ( )		
Alcance:			
Definir el alcance del proceso de verificación, especificando las responsabilidades que deben asumir los roles involucrados**.**			
Objetivos:			
Describir los objetivos del plan.			
Documentos Relacionados			
Documentos usados para elaborar el Plan de Verificación Ej.: • Especificación de Requerimientos			
Requerimientos			
Elementos, casos de uso, requerimientos funcionales y requerimientos no funcionales, que serán verificados. Ej.: • Registrarse (Usuario no registrado) • Loguearse (Usuario no logueado) • Editar Perfil de usuario (Usuario logueado) • Ver Perfil de un autor (Usuario en general) Requerimientos no funcionales que serán verificados a lo largo de todo el proyecto (podrán modificarse en el futuro) • Se podrá acceder a dicha aplicación desde los siguientes browsers: Internet Explorer, etc. • El tiempo estimado de visualización es de 512 Kbits/segundos.			

Listado de Casos de Pruebas por prioridad de Caso de Uso

Prioridad	Caso de Uso	ID Caso de Uso	Caso de Prueba	ID Caso de Prueba
1	Autenticar Usuarios	CU001	Seguridad y control de acceso.	CP001

Modelo de Caso de Prueba

Se describirán en detalle cada uno de los casos de pruebas que se hayan identificado como necesarios para verificar la funcionalidad completa del sistema. Se deberá repetir una tabla por cada caso de prueba que se defina.

<Nombre caso prueba>			**<ID del CP>**
ID Caso de Uso			
Autor del Caso de Prueba	*<Persona que elabora el caso de prueba>*		
Nombre del Probador	*<Persona que ejecuta la prueba>*		
Fecha de Creación:		Fecha de Ejecución:	
Versión:			
Descripción			
<Descripción del caso de prueba>			
Prerrequisitos			
<Enumerar los prerrequisitos para la prueba>			
Pasos:			
<Pasos generales para la prueba, basados en los escenarios de los casos de uso, si existen.>			
Resultado esperado:			
<Resultado esperado de la prueba>			
Resultado obtenido:			
<Resultado obtenido de la ejecución del caso de prueba>			
Decisión de Aprobación del Caso de Prueba			
Aprobó ☐		Falló ☐	
Firma del Probador		Firma del Área Usuaria	
Fecha de Aprobación del Caso de Prueba:			

Anexo N° 21: Formato "Implantación del sistema" (Anexo 25 – Obs. 3)

Municipalidad Provincial de Chiclayo	FORMATO "**IMPLANTACIÓN** LACIÓN DEL SISTEMA"		
	Código: IS	**Versión:**	**Año:**

IMPLANTACIÓN DEL SISTEMA

Objetivo:	Indicar el objetivo del documento
Alcance:	Una breve descripción del alcance de este documento, qué otro(s) sistema(s) están asociados o se ven afectados por este documento
Configuración de Sistemas	Esta sección contiene todo lo necesario para verificar la integridad de la aplicación y de la base de datos
Definiciones y abreviaciones	Esta sección brinda la definición de aquellos términos y abreviaciones requeridas para interpretar adecuadamente el contenido de este documento.

Despliegue del sistema

Coloque aquí una explicación detallada de los pasos necesarios o consideraciones a tener en cuenta para realizar el correcto despliegue del pase.

DETALLE DEL PASE			
SISTEMA / APLICATIVO		**N° DE REQUERIMIENTO DE PASE**	**[Nº de RQP]**
CÓDIGO			
ACRÓNIMO			
GERENCIA/DIVISIÓN RESPONSABLE			

DESCRIPCIÓN DEL REQUERIMIENTO
Descripción del requerimiento que se está implementando.

BASE DE DATOS				
PLAN DE COMPÍLACIÓN DE BD O NOMBRE DE SCRIPT	**# BUILD**	**TIPO**	**VERSIÓN**	**REPOSITORIO**
[Cod. Proyecto] - Plan [Nombre de proyecto a desplegar]_Ejecución_[Tipo de Pase]_BD	[# de build]	Ejm: DDL		BAMBOO

APLICACIÓN			
PLAN DE COMPILACIÓN DE LA APLICACIÓN O NOMBRE DE WAR	**# BUILD**	**TIPO**	**REPOSITORIO**

[Cod. Proyecto] - Plan [Nombre de proyecto a desplegar]_Ejecución_de_Aplicación	[Colocar el # de build]	Ejm: WAR	BAMBOO
[Cod. Proyecto] - Plan [Nombre de proyecto a desplegar]_Ejecución_de_Aplicación	[Colocar el # de build]	EAR	BAMBOO

EJECUCIÓN DE DESPLIEGUE

[Indicar la secuencia de pasos a seguir para la ejecución del despliegue en ambiente de Certificación o Producción especificando el orden de ejecución de cada objeto o archivo mencionado y las acciones a seguir.

PASOS	DESCRIPCIÓN
1	**(1)** Ejecutar el [nombre de script] en el [nombre de esquema de bd] de la [Base de Datos]
2	**(1)** Ejecutar el [Cod. Proyecto] - Plan [Nombre de proyecto a desplegar]_Ejecución_Pase_BD - Build #[Colocar el # de build]
3	**(1)** Ejecutar el [Cod. Proyecto] - Plan Nombre de proyecto a desplegar]_Ejecución_Datos_BD - Build #[Colocar el # de build]
4	**(1)** Plan [Cod. Proyecto] - Nombre de proyecto a desplegar]_Ejecución_de Aplicación - Build # [Colocar el # de build]

(1) Incluir solo para requerimientos que se son atendidos a través de **xxxxxxx.**

PROCEDIMIENTO DE ROLLBACK

[Indicar cuál es la secuencia a seguir para el proceso de la reversión del aplicativo en caso que falle en producción, se debe de incluir los cambios en el ejecutable así como para la base de datos].

Pasos	Descripción
1	
2	
3	

REGISTRO EN MÓDULO DE SEGURIDAD (En caso aplique)

[Indicar los datos del aplicativo o módulos para su inscripción en el Sistema de Seguridad de Usuarios, esto sólo aplica cuando es un nuevo sistema o módulo. Por ejemplo:]

APLICATIVO/MÓDULO	ACTION	JSP	PERMISOS
Módulo Archivo Instrucción Preliminar	archivoInstruccion.do?method=inicio	busquedaArchivoInstruccion.jsp	IN,MO,CO,EL
Reporte Archivo Instrucción Preliminar	reporteAIP.do?method=inicio	reporteAIP.jsp	CO

Recomendaciones:

Coloque aquí las recomendaciones que crea conveniente.

Anexo N° 22: Acta de Cierre

ACTA DE CIERRE DEL MÓDULO DE _____

El día _____, el Jefe del Área usuaria de _____ y el Jefe de Sistemas _____por parte de la Municipalidad de Chiclayo, firman el acta de cierre del módulo de _____ con las siguientes características técnicas y especificaciones como se detalla a continuación:

1. **DETALLES DE LA ENTREGA DEL MODULO**
   1.1 ALCANCE DEL MODULO
   1.2 ETAPAS DE DESARROLLO
   Para cumplir con las etapas de desarrollo del módulo, se realizaron ___ (número) reunión(es) en la cual participaron parte usuaria del área de _____y el equipo de desarrollo del Área de Sistemas de la Municipalidad Provincial de Chiclayo.

ETAPAS	CUMPLIMIENTO (%)
REQUERIMIENTO DE USUARIO (requisitos de diseño)	
CODIFICACION	
PUESTA EN PRODUCCION	
IMPLEMENTACION ASISTIDA	

2. **CAPACITACIÓN**
   -Acta de conformidad

3. **SERVICIOS POS PRODUCCION**
   La MPCH asume el compromiso de atender los futuros servicios de actualización y soporte que el usuario requiera. Estos servicios serán atendidos dependiendo la complejidad, la disponibilidad de recursos y de factores regulatorios.

Por consiguiente, ambas partes representativas aceptan la presente acta dando conformidad al cierre del módulo de _____ en cumplimiento parcial al contrato.

_____	_____
Jefe de Sistemas	Jefe del Área usuaria
POR LA ENTREGA	POR LA RECEPCIÓN

Anexo N° 23: Ejemplo de estimación de tiempo con el método puntos de casos de uso.

El ejemplo práctico a través del cual se describirá el método consiste en una aplicación Web para la gestión de información telefónica, de reportes de averías, y de estadísticas de los estados de los teléfonos de un Centro de Educación Superior (CES). A continuación, se presentan los actores y casos de uso identificados.

Nombre del actor	Casos de Uso Asociados
Usuario anónimo	Insertar Avería, Efectuar Búsquedas
Usuario avanzado	Gestionar Asignar teléfono. Cambiar Contraseña Enviar notificación
Administrador	Gestionar Avería Gestionar Seguimiento de Avería Gestionar Área, Local, Subárea, Teléfonos, Tipos de teléfono, Estado. Gestionar Usuario, Gestionar rol.

PROCEDIMIENTO DE CÁLCULO DE LOS PUNTOS DE CASOS DE USO

A continuación, se van a explicar los pasos generales para el cálculo de puntos casos de uso

1. **Calcular los puntos casos de uso sin ajustar:**

   Para realizar el cálculo se tiene que realizar los siguientes pasos:

1.1 Clasificar cada interacción entre actor y caso de uso según su complejidad y asignar un peso en función de ésta (UAW)

Tipo de interacción	Descripción	Factor de peso	N° de actores	Resultado
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Aplication Programming Interface	1	0	0
Promedio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	0	0
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	3	9
			Total UAW=	**9**

1.2 Calcular la complejidad de caso de uso según número de transacciones o pasos del mismo: (UUCW)

Tipo de caso de uso	Descripción	Factor de peso	Número de Casos de Uso	Resultado
Simple	1-3 Transacciones	5	8	40
Promedio	4-7 Transacciones	10	9	90
Complejo	Mayor de 8 Transacciones.	15	2	30
			Total UUCW=	**160**

1.3 Calcular los puntos casos de uso sin ajustar

Para obtener los puntos de casos de usos sin ajustar utilizaremos la siguiente fórmula:

UUCP = 9 + 160=169

II) Calcular los puntos casos de uso ajustados:

a) Cálculo de los factores técnicos (TCF)

Número de factor	Descripción	Comentario	Peso (W)	Valor (V)	Factor (W*V)
T1	Sistema Distribuido	El sistema es Web, por lo que posee cierto nivel de distribución	2	1	2
T2	Tiempo de respuesta	El tiempo de respuesta respalda los objetivos que se persiguen con el proyecto realizado, por lo que es el adecuado.	1	1	1
T3	Eficiencia por el usuario	Algunos roles necesitan estar relacionados con el sistema para su mejor funcionamiento.	1	3	3
T4	Proceso interno complejo	El sistema no posee cálculos complejos, aunque proporciona una serie de datos lógicos que necesitan un nivel medio de conocimiento para lograr su correcta comprensión.	1	3	3
T5	Reusabilidad	No es objetivo esencial hacer reusabilidad del código, a pesar de que este será orientado a objetos y podrá ser usado por sistemas similares.	1	2	2
T6	Facilidad de instalación	Por ser un sistema Web la complejidad de instalación es mínima.	0.5	1	0.5
T7	Facilidad de uso	El sistema debe ser fácil de usar, aunque se encuentra dirigido a personas ajenas al centro, además.	0.5	5	2.5
T8	Portabilidad	El sistema se encuentra diseñado para que sea usado en situaciones similares en otras empresas, además como está desarrollado en .Net puede ser publicado en cualquier plataforma.	2	5	10
T9	Facilidad de cambio	El sistema encuentra estructurada para que los cambios realizados afecten lo menos posible las funcionalidades del sistema.	1	5	
T10	Concurrencia	La concurrencia es tratada con suma importancia.	1	5	5

T11	Objetivos especiales de seguridad	La seguridad del sistema es un tema bastante controlado, ya que el sistema sólo permite que un usuario realice las funcionalidades correspondientes a su rol dentro del sitio.	1	5	5
T12	Acceso directo a terceras partes	La aplicación es accesible a cualquier usuario.	1	2	2
T13	Facilidades especiales de entrenamiento a usuarios finales	No se hace necesario el entrenamiento de los usuarios finales, debido a la facilidad de uso que presenta el sistema, pero se debe incluir un manual de usuario para garantizar la correcta usabilidad de dicho sistema.	1	1	1
			Total =		42

Con la siguiente fórmula se calcula el factor técnico que aplicaremos en la fórmula final para calcular los puntos de casos de uso ajustados:

TCF=0.6+0.01*42=1.02

b) Cálculo de los factores de entorno (EF)

Número del factor	**Descripción**	**Comentario**	**Peso (W)**	**Valor (V)**	**Factor (W*V)**
E1	Familiaridad con el modelo del proyecto usado.	Se está familiarizado con el modelo del proyecto, pero la experiencia en el modelado es media.	1.5	3	4.5
E2	Experiencia en la aplicación	No es una aplicación que requiera de mucha experiencia, pero se necesita de un equipo capacitado y de conocimientos suficientes para garantizar su correcto funcionamiento.	0.5	4	2
E3	Experiencia OO.	Se considera cierto grado de experiencia en la programación orientada a objetos (OO), debido a que esta es la que se ha estudiado y trabajado.	1	4	4
E4	Capacidad del analista líder.	No existe analista líder, los analistas que integran el equipo de trabajo poseen capacidad media.	0.5	3	1.5
E5	Motivación.	Alta	1	5	5
E6	Estabilidad de los	Aunque el sistema se encuentra sujeto a cambios, el mismo brinda las	2	4	6

	requerimientos.	funcionalidades esenciales que dan cumplimiento a los objetivos que iniciaron su realización.			
E7	Personal media jornada.	Se trabajará a tiempo completo.	-1	0	0
E8	Dificultad en lenguaje de programación.	Como el lenguaje empleado fue C# y este ofrece grandes facilidades y ventajas, se considera una dificultad media su empleo.	-1	3	-3
			Total =		22

Con la siguiente fórmula se calcula el factor entorno que aplicaremos en la fórmula final para calcular los puntos de casos de uso ajustados:

EF=1.4+0.03*22=0.74

c) Calcular los puntos casos de uso sin ajustar

Para obtener los puntos de casos de usos ajustados utilizaremos la siguiente fórmula:

UCP = 169 * 1.02 * 0.74=127.56

PROCEDIMIENTO DE CÁLCULO DEL ESFUERZO

El esfuerzo en horas-hombre viene dado por:

E = 127.56 * 20=2551.2 --- > (Tiempo estimado para la programación)

E: esfuerzo estimado en horas-hombre.

UCP: Puntos de casos de uso ajustados.

PF: Factor de conversión (20 horas-hombre por defecto).

Distribución genérica del esfuerzo.

Actividad	Porcentaje
Análisis	10 %
Diseño	20 %
Programación	40 %
Pruebas	15 %
Sobrecarga (otras actividades)	15 %

Con este criterio y tomando como entrada la estimación de tiempo calculada a partir de los Puntos de Casos de Uso, se pueden calcular las demás estimaciones para obtener la duración total del proyecto.

Distribución real del esfuerzo.

Análisis **40 % → 2551.2** **10 % → X** **X=** (2551.2*10%) / 40% **X=637.8**	**Diseño** **40 % → 2551.2** **20 % → X** **X=** (2551.2*20%) / 40% **X=1275.6**
Pruebas **40 % → 2551.2** **15 % → X** **X=** (2551.2*15%) / 40% **X=956.7**	**Sobrecarga** **40 % → 2551.2** **15 % → X** **X=** (2551.2*15%) / 40% **X=956.7**

Actividad	Horas-Hombre
Análisis	637.8
Diseño	1275.6
Programación	2551.2
Pruebas	956.7
Sobrecarga (otras actividades)	956.7

Total	**6378**

Cálculo del esfuerzo total:

ETotal = 6378 horas /hombre

Cálculo del tiempo de desarrollo:

TDesarrollo = ETotal/CHTotal CHTotal: Cantidad de hombres

TDesarrollo = 6378/2

TDesarrollo = 3189 horas

Considerando que se trabajan 8 horas diarias:

TDesarrollo = TDesarrollo/8 horas/día

TDesarrollo = 3189 horas/8 horas/día

TDesarrollo = 399 días aproximadamente

Anexo N° 24: Formato para validación de expertos del método propuesto

Estimado Ingeniero:

A través de la presente nos dirigimos a usted con el fin de solicitarle su ayuda para la validación de la propuesta realizada en la investigación denominada **MÉTODO PARA GESTIÓN DE DESARROLLO DE SOFTWARE BASADO EN ISO 12207:2017 PARA LA MUNICIPALIDAD PROVINCIAL DE CHICLAYO**. Para tal fin, se anexa el cuestionario de validación.

Agradecemos su valiosa colaboración.

NOMBRES Y APELLIDOS	: OSCAR GILBERTO ZOCON ALVA
FORMACIÓN ACADÉMICA	: INGENIERO DE COMPUTACIÓN Y SISTEMAS
AREAS DE EXPERIENCIA	: DESARROLLO DE PROYECTOS,
PROFESIONAL	: GERENCIA DE TI
TIEMPO DE EXPERIENCIA	: 22 AÑOS
CARGO ACTUAL	: GERENTE GENERAL
INSTITUCIÓN	: SOLUCIONES GLOBALES EMPRESARIALES
CARGO	: PRESIDENTE DEL CAPITULO DE SISTEMAS, COMPUTACIÓN E INFORMÁTICA
INSTITUCIÓN	: COLEGIO DE INGENIEROS DEL PERÚ – CONSEJO DEPARTAMENTAL DE CAJAMARCA

Objetivo de la investigación	: Desarrollar un método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 que permita mejorar el proceso de desarrollo de software en la Municipalidad Provincial de Chiclayo
Objetivo del juicio de expertos	: Validar el método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 en relación a la funcionalidad, fiabilidad, usabilidad, rendimiento y mantenibilidad.

De acuerdo con los siguientes indicadores califique cada uno de los ítems según corresponda.

CATEGORIA	CALIFICACIÓN	INDICADOR
FUNCIONABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de proveer los resultados, efectos correctos o acordados, con el grado necesario de precisión.	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes obtener los resultados esperados en el producto de software
	2. Bajo Nivel	La actividad o tarea permiten obtener algunos efectos correctos, pero no corresponden con la totalidad de obtener los resultados esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.
	4. Alto nivel	La actividad o tarea de la metodología permiten la correcta funcionalidad del producto de software.
USABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir al usuario aprender el manejo del producto de software operarlo y controlarlo.	1. No cumple con el criterio	La actividad o tarea de la metodología no permite la correcta usabilidad en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para lograr su correcto uso
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr que el usuario maneje opere y controle el producto de software.
	4. Alto nivel	La actividad o tarea permite la correcta usabilidad del producto de software
MANTENIBILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de evitar efectos inesperados a causa de modificar el producto de software	1. No cumple con el criterio	La actividad o tarea no es capaz de soportar modificaciones en el producto de software
	2. Bajo Nivel	La actividad o tarea es poco probable que soporte modificaciones en el producto de software
	3. Moderado nivel	La actividad o tarea acepta efectos inesperados en el producto de software
	4. Alto nivel	La actividad o tarea se encuentra completamente apta para adaptarse a modificaciones ante algún efecto inesperado.
RENDIMIENTO Los aspectos considerados en la actividad o tarea de la metodología son capaces de proporcionar los tiempos de respuesta y procesamiento apropiados, además de cumplir con los requisitos determinados.	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes cumplir con los requisitos determinados.
	2. Bajo Nivel	La actividad o tarea permiten obtener algunos tiempos de respuesta adecuados, pero no corresponden con la totalidad de obtener los requisitos esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.

	4. Alto nivel	La actividad o tarea de la metodología permiten proporcionar tiempos de respuesta y procesamiento adecuados, y cumplir con los requisitos.
CATEGORIA	**CALIFICACIÓN**	**INDICADOR**
FIABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir la operabilidad aun cuando se presenten fallos	1. No cumple con el criterio	La actividad o tarea de la metodología no da fiabilidad a los usuarios en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para que opere normalmente el producto de software.
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr la fiabilidad en la operabilidad del producto de software aun cuando presente fallos.
	4. Alto nivel	La actividad o tarea permite que el producto de software sea fiable para los usuarios al momento de usar el producto de software.

CUESTIONARIO PARA VALIDACIÓN DEL MÉTODO DE GESTIÓN DE DESARROLLO DE SOFTTWARE BASADO EN ISO 12207:2017 PARA LA MUNICIPALIDAD PROVINCIAL DE CHICLAYO

PROCESOS	INDICADOR						OBSERVACIÓN
	FUNCIONALIDAD		RENDIMIENTO		FIABILIDAD		
	¿Cree usted que desde el punto de vista funcional las actividades y tareas considerados en el método propuesto son capaces de proveer los resultados esperados por los usuarios??	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda adherirse a estándares, normas y buenas prácticas relacionadas con funcionalidad??	¿Cree usted que a nivel de rendimiento las actividades y tareas considerados en el método propuesto son capaces de permitir al software cumplir con los requisitos de los usuarios??	¿Cree usted que las actividades y tareas considerados en el método propuesto den la capacidad al software para proporcionar los tiempos de respuesta y procesamiento apropiados??	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda estar operativo y accesible para su uso cuando se necesite y además cumpla con las necesidades de fiabilidad durante el funcionamiento normal??	¿Cree usted que a nivel de fiabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software para operar cuando se presenten fallos?	
Definición de requisitos y necesidades de las partes interesadas	4	4	3	3	4	4	**OBS. 1:** Abarcar un poco más la gestión de cambios a requerimientos, los requerimientos van cambiando.
Definición de requisitos del Sistema de software	4	4	3	3	4	4	Ninguna
Análisis del sistema	3	3	3	3	3	3	Ninguna
Definición de arquitectura del software	4	4	4	4	4	4	Ninguna
Diseño del software	3	3	3	3	3	3	Ninguna
Implementación del software	4	4	4	4	4	4	Ninguna

PROCESOS	INDICADOR						OBSERVACIÓN
	FUNCIONALIDAD		RENDIMIENTO		FIABILIDAD		
	¿Cree usted que desde el punto de vista funcional las actividades y tareas considerados en el método propuesto son capaces de proveer los resultados esperados por los usuarios?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda adherirse a estándares, normas y buenas prácticas relacionadas con funcionalidad??	¿Cree usted que a nivel de rendimiento las actividades y tareas considerados en el método propuesto son capaces de permitir al software cumplir con los requisitos de los usuarios?	¿Cree usted que las actividades y tareas considerados en el método propuesto den la capacidad al software para proporcionar los tiempos de respuesta y procesamiento apropiados?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda estar operativo y accesible para su uso cuando se necesite y además cumpla con las necesidades de fiabilidad durante el funcionamiento normal?	¿Cree usted que a nivel de fiabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software para operar cuando se presenten fallos?	
Integración del software	4	4	3	3	4	4	Ninguna
Verificación del software	3	4	4	4	4	4	Ninguna
Transición del software	3	3	4	4	4	4	**OBS. 2:** En algunos escenarios es necesario la migración de datos, por lo que podría considerarse en alguna etapa
Validación del software	4	4	4	4	4	4	Ninguna
Operación del software	4	3	4	3	4	4	Ninguna
Mantenimiento del software	4	4	4	4	4	3	Ninguna

PROCESOS	INDICADOR				OBSERVACIÓN
	USABILIDAD		MANTENIMIENTO		
	¿Cree usted que a nivel de operabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software de que el usuario pueda usarlo y controlarlo con facilidad?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software tenga la capacidad en que su interfaz llegue a satisfacer y agradar al usuario?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda sufrir una determinada modificación que sea implementada y que esto no afecte a otras funcionalidades del software?	¿Cree usted que a nivel de mantenimiento las actividades y tareas considerados en el método propuesto den la capacidad al software para que pueda ser reusado para crear otro sistema, ser modificado sin causar daños y ser probado para determinar si se han cumplido los requerimientos del usuario?	
Definición de requisitos y necesidades de las partes interesadas	4	4	4	4	Ninguna
Definición de requisitos del Sistema de software	4	4	3	4	Ninguna
Análisis del sistema	3	3	4	4	Ninguna
Definición de arquitectura del software	4	4	3	4	Ninguna
Diseño del software	3	3	4	4	Ninguna
Implementación del software	4	4	4	4	Ninguna

PROCESOS	INDICADOR				OBSERVACIÓN
	USABILIDAD		MANTENIMIENTO		
	¿Cree usted que a nivel de operabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software de que el usuario pueda usarlo y controlarlo con facilidad?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software tenga la capacidad en que su interfaz llegue a satisfacer y agradar al usuario?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda sufrir una determinada modificación que sea implementada y que esto no afecte a otras funcionalidades del software?	¿Cree usted que a nivel de mantenimiento las actividades y tareas considerados en el método propuesto den la capacidad al software para que pueda ser reusado para crear otro sistema, ser modificado sin causar daños y ser probado para determinar si se han cumplido los requerimientos del usuario?	
Integración del software	3	3	4	4	**OBS. 3:** Se podría detallar mejor la Integración.
Verificación del software	4	4	4	4	Ninguna
Transición del software	3	3	3	4	**OBS. 4:** Agregar tareas de Incorporación de Tutoriales, Manuales interactivos.
Validación del software	4	4	3	4	Ninguna
Operación del software	4	4	4	4	Ninguna
Mantenimiento del software	4	3	4	3	**OBS. 5:** Se trabaja un mantenimiento correctivo, pero se puede incorporar un mantenimiento preventivo o de mejora u optimización

Anexo N° 25: Formato para validación de expertos del método propuesto

Estimado Ingeniero:

A través de la presente nos dirigimos a usted con el fin de solicitarle su ayuda para la validación de la propuesta realizada en la investigación denominada **MÉTODO PARA GESTIÓN DE DESARROLLO DE SOFTWARE BASADO EN ISO 12207:2017 PARA LA MUNICIPALIDAD PROVINCIAL DE CHICLAYO**. Para tal fin, se anexa el cuestionario de validación.

Agradecemos su valiosa colaboración.

NOMBRES Y APELLIDOS	: Edwin José Benavente Millán
FORMACIÓN ACADÉMICA	: Ingeniero en Computación e Informática Magister en Gerencia de Tecnologías de la Información y Gestión del Software
AREAS DE EXPERIENCIA	: Ingeniería del Software Arquitectura de Software Gestión de Proyectos
PROFESIONAL	
TIEMPO DE EXPERIENCIA	: 15 años
CARGO ACTUAL	: Arquitecto de Software
INSTITUCIÓN	: Nexteppe Business Solutions

Objetivo de la investigación	: Desarrollar un método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 que permita mejorar el proceso de desarrollo de software en la Municipalidad Provincial de Chiclayo
Objetivo del juicio de expertos	: Validar el método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 en relación a la funcionalidad, fiabilidad, usabilidad, rendimiento y mantenibilidad.

De acuerdo con los siguientes indicadores califique cada uno de los ítems según corresponda.

CATEGORIA	CALIFICACIÓN	INDICADOR
FUNCIONABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de proveer los resultados, efectos correctos o acordados, con el grado necesario de precisión.	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes obtener los resultados esperados en el producto de software
	2. Bajo Nivel	La actividad o tarea permiten obtener algunos efectos correctos, pero no corresponden con la totalidad de obtener los resultados esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.
	4. Alto nivel	La actividad o tarea de la metodología permiten la correcta funcionalidad del producto de software.
USABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir al usuario aprender el manejo del producto de software operarlo y controlarlo.	1. No cumple con el criterio	La actividad o tarea de la metodología no permite la correcta usabilidad en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para lograr su correcto uso
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr que el usuario maneje opere y controle el producto de software.
	4. Alto nivel	La actividad o tarea permite la correcta usabilidad del producto de software
MANTENIBILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de evitar efectos inesperados a causa de modificar el producto de software	1. No cumple con el criterio	La actividad o tarea no es capaz de soportar modificaciones en el producto de software
	2. Bajo Nivel	La actividad o tarea es poco probable que soporte modificaciones en el producto de software
	3. Moderado nivel	La actividad o tarea acepta efectos inesperados en el producto de software
	4. Alto nivel	La actividad o tarea se encuentra completamente apta para adaptarse a modificaciones ante algún efecto inesperado.
RENDIMIENTO Los aspectos considerados en la actividad o tarea de la metodología son capaces de proporcionar los tiempos de respuesta y procesamiento apropiados, además de cumplir con los requisitos determinados.	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes cumplir con los requisitos determinados.
	2. Bajo Nivel	La actividad o tarea permiten obtener algunos tiempos de respuesta adecuados, pero no corresponden con la totalidad de obtener los requisitos esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.
	4. Alto nivel	La actividad o tarea de la metodología permiten proporcionar tiempos de respuesta y procesamiento adecuados, y cumplir con los requisitos.

CATEGORIA	CALIFICACIÓN	INDICADOR
FIABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir la operabilidad aun cuando se presenten fallos	1. No cumple con el criterio	La actividad o tarea de la metodología no da fiabilidad a los usuarios en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para que opere normalmente el producto de software.
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr la fiabilidad en la operabilidad del producto de software aun cuando presente fallos.
	4. Alto nivel	La actividad o tarea permite que el producto de software sea fiable para los usuarios al momento de usar el producto de software.

CUESTIONARIO PARA VALIDACIÓN DEL MÉTODO DE GESTIÓN DE DESARROLLO DE SOFTTWARE BASADO EN ISO 12207:2017 PARA LA MUNICIPALIDAD PROVINCIAL DE CHICLAYO

PROCESOS	INDICADOR						OBSERVACIÓN
	FUNCIONALIDAD		RENDIMIENTO		FIABILIDAD		
	¿Cree usted que desde el punto de vista funcional las actividades y tareas considerados en el método propuesto son capaces de proveer los resultados esperados por los usuarios??	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda adherirse a estándares, normas y buenas prácticas relacionadas con funcionalidad??	¿Cree usted que a nivel de rendimiento las actividades y tareas considerados en el método propuesto son capaces de permitir al software cumplir con los requisitos de los usuarios??	¿Cree usted que las actividades y tareas considerados en el método propuesto den la capacidad al software para proporcionar los tiempos de respuesta y procesamiento apropiados??	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda estar operativo y accesible para su uso cuando se necesite y además cumpla con las necesidades de fiabilidad durante el funcionamiento normal??	¿Cree usted que a nivel de fiabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software para operar cuando se presenten fallos?	
Definición de requisitos y necesidades de las partes interesadas	3	3	3	3	4	4	Ninguna
Definición de requisitos del Sistema de software	3	3	3	3	4	4	**OBS.1:** para el rendimiento se debe calcular intervalos de tiempo en las fases del modelo.
Análisis del sistema	3	3	3	3	3	3	Ninguna
Definición de arquitectura del software	4	4	4	4	4	4	**OBS. 2:** en la actualidad existen parámetros de arquitectura de software que deberían considerarse. (es decir que atributos de calidad se están cumpliendo)

Diseño del software	4	4	4	4	4	4	**OBS. 3:** No se hace diferenciación del tipo de sistema o del modelo que se va a utilizar
Implementación del software	3	3	3	3	3	3	Ninguna

PROCESOS	INDICADOR						OBSERVACIÓN
	FUNCIONALIDAD		RENDIMIENTO		FIABILIDAD		
	¿Cree usted que desde el punto de vista funcional las actividades y tareas considerados en el método propuesto son capaces de proveer los resultados esperados por los usuarios?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda adherirse a estándares, normas y buenas prácticas relacionadas con funcionalidad??	¿Cree usted que a nivel de rendimiento las actividades y tareas considerados en el método propuesto son capaces de permitir al software cumplir con los requisitos de los usuarios?	¿Cree usted que las actividades y tareas considerados en el método propuesto den la capacidad al software para proporcionar los tiempos de respuesta y procesamiento apropiados?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda estar operativo y accesible para su uso cuando se necesite y además cumpla con las necesidades de fiabilidad durante el funcionamiento normal?	¿Cree usted que a nivel de fiabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software para operar cuando se presenten fallos?	
Integración del software	3	4	4	3	3	4	Ninguna
Verificación del software	3	4	4	3	4	3	Ninguna
Transición del software	3	4	4	4	3	4	Ninguna
Validación del software	4	3	4	3	4	4	Ninguna
Operación del software	3	3	3	3	3	3	Ninguna
Mantenimiento del software	3	3	3	3	3	3	Ninguna

PROCESOS	INDICADOR				OBSERVACIÓN
	USABILIDAD		MANTENIMIENTO		
	¿Cree usted que a nivel de operabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software de que el usuario pueda usarlo y controlarlo con facilidad?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software tenga la capacidad en que su interfaz llegue a satisfacer y agradar al usuario?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda sufrir una determinada modificación que sea implementada y que esto no afecte a otras funcionalidades del software?	¿Cree usted que a nivel de mantenimiento las actividades y tareas considerados en el método propuesto den la capacidad al software para que pueda ser reusado para crear otro sistema, ser modificado sin causar daños y ser probado para determinar si se han cumplido los requerimientos del usuario?	
Definición de requisitos y necesidades de las partes interesadas	4	3	4	3	Ninguna
Definición de requisitos del Sistema de software	4	4	4	3	Ninguna
Análisis del sistema	4	4	4	4	Ninguna
Definición de arquitectura del software	4	3	4	4	**OBS. 4:** Considerar criterios de usabilidad y mantenimiento para la arquitectura del software dentro del modelo.
Diseño del software	4	4	4	3	**OBS. 5:** considerar que características específicas del usuario se van a plasmar dentro del modelo (accesibilidad, capacidad de aprendizaje del usuario)
Implementación del software	4	4	4	4	Ninguna

PROCESOS	INDICADOR				OBSERVACIÓN
	USABILIDAD		MANTENIMIENTO		
	¿Cree usted que a nivel de operabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software de que el usuario pueda usarlo y controlarlo con facilidad?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software tenga la capacidad en que su interfaz llegue a satisfacer y agradar al usuario?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda sufrir una determinada modificación que sea implementada y que esto no afecte a otras funcionalidades del software?	¿Cree usted que a nivel de mantenimiento las actividades y tareas considerados en el método propuesto den la capacidad al software para que pueda ser reusado para crear otro sistema, ser modificado sin causar daños y ser probado para determinar si se han cumplido los requerimientos del usuario?	
Integración del software	4	4	4	4	Ninguna
Verificación del software	4	3	3	4	Ninguna
Transición del software	4	4	4	4	Ninguna
Validación del software	4	3	4	4	Ninguna
Operación del software	4	3	4	4	Ninguna
Mantenimiento del software	4	4	4	4	Ninguna

Anexo N° 26: Formato para validación de expertos del método propuesto

Estimado Ingeniero:

A través de la presente nos dirigimos a usted con el fin de solicitarle su ayuda para la validación de la propuesta realizada en la investigación denominada **MÉTODO PARA GESTIÓN DE DESARROLLO DE SOFTWARE BASADO EN ISO 12207:2017 PARA LA MUNICIPALIDAD PROVINCIAL DE CHICLAYO**. Para tal fin, se anexa el cuestionario de validación.

Agradecemos su valiosa colaboración.

NOMBRES Y APELLIDOS	: Junior Eugenio Cachay Maco
FORMACIÓN ACADÉMICA	: Ing. en Computación e Informática
AREAS DE EXPERIENCIA	: Auditoría de sistemas de información
PROFESIONAL	: Ingeniería de software
TIEMPO DE EXPERIENCIA	: 12 AÑOS
CARGO ACTUAL	: Gerente general
INSTITUCIÓN	: AUDIT AND CONTROL OF INFORMATION SYSTEMS

Objetivo de la investigación	: Desarrollar un método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 que permita mejorar el proceso de desarrollo de software en la Municipalidad Provincial de Chiclayo
Objetivo del juicio de expertos	: Validar el método para la gestión del desarrollo del software, basada en la NTP ISO 12207:2017 en relación a la funcionalidad, fiabilidad, usabilidad, rendimiento y mantenibilidad.

De acuerdo con los siguientes indicadores califique cada uno de los ítems según corresponda.

CATEGORIA	CALIFICACIÓN	INDICADOR
FUNCIONABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de proveer los resultados, efectos correctos o acordados, con el grado necesario de precisión.	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes obtener los resultados esperados en el producto de software
	2. Bajo Nivel	La actividad o tarea permiten obtener algunos efectos correctos, pero no corresponden con la totalidad de obtener los resultados esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.
	4. Alto nivel	La actividad o tarea de la metodología permiten la correcta funcionalidad del producto de software.
USABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir al usuario aprender el manejo del producto de software operarlo y controlarlo.	1. No cumple con el criterio	La actividad o tarea de la metodología no permite la correcta usabilidad en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para lograr su correcto uso
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr que el usuario maneje opere y controle el producto de software.
	4. Alto nivel	La actividad o tarea permite la correcta usabilidad del producto de software
MANTENIBILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de evitar efectos inesperados a causa de modificar el producto de software	1. No cumple con el criterio	La actividad o tarea no es capaz de soportar modificaciones en el producto de software
	2. Bajo Nivel	La actividad o tarea es poco probable que soporte modificaciones en el producto de software
	3. Moderado nivel	La actividad o tarea acepta efectos inesperados en el producto de software
	4. Alto nivel	La actividad o tarea se encuentra completamente apta para adaptarse a modificaciones ante algún efecto inesperado.
RENDIMIENTO Los aspectos considerados en la actividad o tarea de la metodología son capaces de proporcionar los tiempos de respuesta y procesamiento apropiados, además de cumplir con los requisitos determinados.	1. No cumple con el criterio	La actividad o tarea de la metodología no son suficientes cumplir con los requisitos determinados.
	2. Bajo Nivel	La actividad o tarea permiten obtener algunos tiempos de respuesta adecuados, pero no corresponden con la totalidad de obtener los requisitos esperados en el producto de software
	3. Moderado nivel	La actividad o tarea debe incrementar algunos aspectos para poder obtener los resultados esperados en el producto de software completamente.
	4. Alto nivel	La actividad o tarea de la metodología permiten proporcionar tiempos de respuesta y procesamiento adecuados, y cumplir con los requisitos.

CATEGORIA	CALIFICACIÓN	INDICADOR
FIABILIDAD Los aspectos considerados en la actividad o tarea de la metodología son capaces de permitir la operabilidad aun cuando se presenten fallos	1. No cumple con el criterio	La actividad o tarea de la metodología no da fiabilidad a los usuarios en el producto de software
	2. Bajo Nivel	La actividad o tarea requiere significativas modificaciones para que opere normalmente el producto de software.
	3. Moderado nivel	Se requiere modificaciones de algunos aspectos de la actividad o tarea para lograr la fiabilidad en la operabilidad del producto de software aun cuando presente fallos.
	4. Alto nivel	La actividad o tarea permite que el producto de software sea fiable para los usuarios al momento de usar el producto de software.

CUESTIONARIO PARA VALIDACIÓN DEL MÉTODO DE GESTIÓN DE DESARROLLO DE SOFTTWARE BASADO EN ISO 12207:2017 PARA LA MUNICIPALIDAD PROVINCIAL DE CHICLAYO

PROCESOS	INDICADOR						OBSERVACIÓN
	FUNCIONALIDAD		RENDIMIENTO		FIABILIDAD		
	¿Cree usted que desde el punto de vista funcional las actividades y tareas considerados en el método propuesto son capaces de proveer los resultados esperados por los usuarios??	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda adherirse a estándares, normas y buenas prácticas relacionadas con funcionalidad??	¿Cree usted que a nivel de rendimiento las actividades y tareas considerados en el método propuesto son capaces de permitir al software cumplir con los requisitos de los usuarios??	¿Cree usted que las actividades y tareas considerados en el método propuesto den la capacidad al software para proporcionar los tiempos de respuesta y procesamiento apropiados??	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda estar operativo y accesible para su uso cuando se necesite y además cumpla con las necesidades de fiabilidad durante el funcionamiento normal??	¿Cree usted que a nivel de fiabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software para operar cuando se presenten fallos?	
Definición de requisitos y necesidades de las partes interesadas	3	3	3	3	3	3	**OBS.1:** Consideraría como tarea adicional, analizar a qué objetivo estratégico empresarial apoyan dichos requisitos.
Definición de requisitos del Sistema de software	4	4	3	3	3	3	**OBS. 2:** Sugiero considerar estimaciones horas hombre tanto para implementar como probar
Análisis del sistema	3	3	3	3	3	3	Ninguna.
Definición de arquitectura del software	3	3	3	3	3	3	Ninguna.
Diseño de software	3	3	3	3	3	3	Ninguna.
Implementación del software	3	3	3	3	3	3	Ninguna.

PROCESOS	INDICADOR						OBSERVACIÓN
	FUNCIONALIDAD		RENDIMIENTO		FIABILIDAD		
	¿Cree usted que desde el punto de vista funcional las actividades y tareas considerados en el método propuesto son capaces de proveer los resultados esperados por los usuarios?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda adherirse a estándares, normas y buenas prácticas relacionadas con funcionalidad??	¿Cree usted que a nivel de rendimiento las actividades y tareas considerados en el método propuesto son capaces de permitir al software cumplir con los requisitos de los usuarios?	¿Cree usted que las actividades y tareas considerados en el método propuesto den la capacidad al software para proporcionar los tiempos de respuesta y procesamiento apropiados?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda estar operativo y accesible para su uso cuando se necesite y además cumpla con las necesidades de fiabilidad durante el funcionamiento normal?	¿Cree usted que a nivel de fiabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software para operar cuando se presenten fallos?	
Integración del software	3	3	3	3	3	3	**OBS. 3:** Sugiero considerar alguna tarea de análisis de automatización de actividades de despliegue
Verificación del software	3	3	3	3	3	3	Ninguna.
Transición del software	3	3	3	3	3	3	Ninguna.
Validación del software	3	3	3	3	3	3	Ninguna.
Operación del software	3	3	3	3	3	3	Ninguna.
Mantenimiento del software	3	3	3	3	3	3	Ninguna.

PROCESOS	INDICADOR				OBSERVACIÓN
	USABILIDAD		MANTENIMIENTO		
	¿Cree usted que a nivel de operabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software de que el usuario pueda usarlo y controlarlo con facilidad?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software tenga la capacidad en que su interfaz llegue a satisfacer y agradar al usuario?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda sufrir una determinada modificación que sea implementada y que esto no afecte a otras funcionalidades del software?	¿Cree usted que a nivel de mantenimiento las actividades y tareas considerados en el método propuesto den la capacidad al software para que pueda ser reusado para crear otro sistema, ser modificado sin causar daños y ser probado para determinar si se han cumplido los requerimientos del usuario?	
Definición de requisitos y necesidades de las partes interesadas	3	3	3	3	ninguna
Definición de requisitos del Sistema de software	3	3	3	3	Ninguna.
Análisis del sistema	3	3	4	3	Ninguna.
Definición de arquitectura del software	3	3	4	4	Ninguna.
Diseño del software	3	3	3	3	Ninguna.
Implementación del software	4	4	4	4	**OBS. 4:** Sugiero considerar control de versiones de código fuente.

PROCESOS	INDICADOR				OBSERVACIÓN
	USABILIDAD		MANTENIMIENTO		
	¿Cree usted que a nivel de operabilidad las actividades y tareas considerados en el método propuesto brindan la capacidad al software de que el usuario pueda usarlo y controlarlo con facilidad?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software tenga la capacidad en que su interfaz llegue a satisfacer y agradar al usuario?	¿Cree usted que las actividades y tareas considerados en el método propuesto permiten que el software pueda sufrir una determinada modificación que sea implementada y que esto no afecte a otras funcionalidades del software?	¿Cree usted que a nivel de mantenimiento las actividades y tareas considerados en el método propuesto den la capacidad al software para que pueda ser reusado para crear otro sistema, ser modificado sin causar daños y ser probado para determinar si se han cumplido los requerimientos del usuario?	
Integración del software	3	3	3	3	Ninguna.
Verificación del software	3	3	3	3	Ninguna.
Transición del software	3	3	4	4	Ninguna.
Validación del software	3	3	3	3	Ninguna.
Operación del software	3	3	3	3	Ninguna.
Mantenimiento del software	3	3	4	4	Ninguna.