DECANATO DE INGENIERÍA E INFORMÁTICA

ESCUELA DE INFORMÁTICA

ANÁLISIS DEL ESTADO ACTUAL DEL PROCESO DE CALIDAD DE SOFTWARE DE LA FACTORÍA DE SOFTWARE WEPSYS S.R.L., SANTO DOMINGO, REP. DOM.

PROYECTO DE TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE:

INGENIERO DE SOFTWARE

SUSTENTANTES:

Br. Carla Milenis Gómez Pérez 2017-0012

Br. Luis José Javier Tatis 2017-0259

ASESOR:

Ing. Luis Núñez

Santo Domingo, Distrito Nacional,
República Dominicana,
Noviembre 2020

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

AGRADECIMIENTOS

A Dios, por realizar su perfecta voluntad en mí y permitirme culminar esta etapa de mi vida. Sé que el mismo Dios que me ha dotado de sentido, razón e intelecto tuvo la intención de que no me abstenga de utilizarlo ¡Soli Deo Gloria!

A mis padres, Manuel Gómez y Milenis Pérez, por inculcarme los principios que me convirtieron en la persona que soy hoy, por apoyar mis sueños y por educarme.

A mis amigos, Anny Gónzalez, Ediandry López, Dariyank Sanquintín y Michael Peralta, por estar a mi lado durante mi carrera universitaria, darme palabras de aliento y por permitirme ver la vida desde un lado positivo, los admiro a todos y a cada uno de ustedes.

A un grupo especial de compañeros que conocí en esta aventura universitaria, “Los coronamigos”, jóvenes con muchas cualidades positivas que me inspiraron a dar lo mejor de mí.

A mi compañero de tesis, Luis Javier, por ser una persona colaboradora y un gran compañero de equipo, siempre dispuesto a escuchar diferentes puntos de vista y examinarlos con pensamiento crítico.

A nuestro asesor, el maestro Luis Núñez, por ser nuestra guía en este trabajo de investigación, por siempre estar disponible y por motivarnos a dar lo mejor con su disciplina, organización y entrega.

A todos los maestros que tuve a lo largo de la carrera, quienes impartieron sus conocimientos y me brindaron la oportunidad de tener una buena formación en el área.

A la universidad APEC, mi casa de estudios, la cual me ha otorgado los recursos y las herramientas necesarias para ser un ente que contribuye a la sociedad y que tiene la oportunidad de hacer cambios importantes con los conocimientos adquiridos.

A todas las personas que de una u otra manera impactaron mi vida, gracias porque directa o indirectamente me ayudaron a convertirme en la persona que soy hoy.

Carla Milenis Gómez Pérez

A Dios, por permitirme culminar este ciclo de mi vida, brindándome la fuerza necesaria para continuar por el camino correcto.

A mis padres, Narda G. Tatis Taveras y Luis D. Javier Amezquita, por estar siempre a mi lado dándome amor y comprensión de manera incondicional. Sin ellos no sería la persona con principios y valores que actualmente soy.

A mis hermanos, Luis A. Javier Tatis y Karla Gisselle Javier Tatis, por darme su apoyo incondicional en todo lo que he realizado, tanto en las buenas como en las malas, los quiero mucho, aunque nunca se los diga.

A mi esposa, Rosmery Patricia Peña Peña, a quien conocí en la universidad APEC y me ha ayudado de todas las maneras posibles a lograr este gran reto. sin ella esto no sería posible. Te amo amor.

A Carla Gómez, mi compañera de tesis, por soportarme a lo largo de cuatro años y aun así comprometerse conmigo a realizar este trabajo de grado, en el cual como siempre ha sido una compañera dispuesta y dedicada a lo que quisimos plasmar en este trabajo.

A mis jefes, Ing. Stalin Rivas, Ing. William Werner y el Ing. Pablo Lorenzo, por darme la oportunidad, creer y confiar en mi cuando nadie más siquiera lo consideró. Me enseñaron cosas que no se aprenden solo trabajando y no hay manera de cómo pagarles ese conocimiento.

A mis amigos y compañeros de clases, especialmente a Manuel Oleaga, Danyer Domínguez, Ezequiel Rodríguez, Víctor Garcés, Ángel Virginio Torres, Diana Rodríguez y Richard Reyes, por brindarme su ayuda y amistad en todo este tiempo.

A mi asesor de tesis, Ing. Luis Núñez, quien desde el primer día se mostró dispuesto a apoyarnos en este largo camino. Gracias por su dedicación y aún más por su organización, ya que fue de una ayuda inmensa.

A cada uno de los profesores que me impartieron clases durante la carrera, quienes me ayudaron en cada una de mis inquietudes, otorgando respuestas contundentes, las cuales me han permitido tener los conocimientos que tengo ahora.

A la Universidad APEC, por hacerme sentir como en mi casa y darme las herramientas que hoy me serán suficientes para lograr mis objetivos.

A todas aquellas personas que de una manera u otra manera confiaron en mí y colaboraron para que hoy esté aquí creciendo como persona; Gracias de corazón.

Luis José Javier Tatis

DEDICATORIA

A mis padres, quienes fueron mis primeros maestros en la vida, gracias por darme educación e inculcarme valores y principios; en la vida hay muchas cosas que cambian, como las opiniones y la manera de pensar, pero los principios permanecen para siempre, este no es el fruto de mi esfuerzo, sino del de ustedes.

Carla Milenis Gómez Pérez

A mi madre Narda, quien siempre se esforzó por darme la mejor educación y que nunca se rindió en su labor de hacerme una persona de bien. Gracias por cada día que te levantaste para hacerme comida. Gracias por cada día que me hiciste saber lo grande que soy. Gracias por cada día que me enseñaste lo que valgo. Gracias por cada día que te tengo a mi lado. Gracias por hacer quien soy. ¡TE AMO MAMÁ! Esto es gracias a ti y para ti.

Luis José Javier Tatis

RESUMEN EJECUTIVO

El presente trabajo de grado engloba el análisis del estado actual del proceso de calidad de software de la empresa Wepsys S.R.L. y la propuesta de mejora del mismo, utilizando prácticas recomendadas por el Modelo de Capacidad y Madurez Integrado (CMMI).

Este análisis ha sido elaborado con el objetivo de conocer la situación actual del proceso de calidad de software, descubrir cuáles son sus debilidades y de esa manera presentar una propuesta de mejora del proceso que se sigue actualmente en la empresa. El nuevo proceso permitirá que se lleven a cabo actividades de aseguramiento de la calidad más completas, que se ahorre tiempo en el ciclo de vida del desarrollo de software, que se ahorren costos monetarios y que se logre un mayor nivel de satisfacción en los clientes que adquieren los servicios de Wepsys, de esta manera, la empresa podría ampliar su presencia en el mercado competitivo local e internacional.

Actualmente la empresa Wepsys cuenta con un proceso de calidad de software nuevo, el cual nunca ha sido evaluado y no se encuentra planteado de manera escrita, sino que, simplemente se realizan las actividades necesarias para cubrir este proceso, esto provoca que cuando se van a realizar las actividades de aseguramiento de calidad por primera vez en un proyecto, la curva de aprendizaje del personal de calidad sea más larga, las actividades realizadas sean más propensas a no realizarse de manera óptima, se queden inconformidades sin resolver y se tarde más tiempo del esperado por el cliente en el ciclo de vida del desarrollo de software porque se encuentran defectos en los demos de aceptación.

Para dar validez a esta problemática, se implementó el método cuantitativo, por medio de la realización de una encuesta con el propósito de conocer la percepción de los colaboradores de las áreas de desarrollo y de gestión de proyectos con relación al proceso de aseguramiento de calidad de software actual. La encuesta fue aplicada al 100% de la población que componen las áreas anteriormente mencionadas.

La encuesta está compuesta por un total de 24 preguntas divididas en dos secciones, la primera sección consta de 10 preguntas que tienen como objetivo conocer informaciones generales y la actividad principal de Wepsys, la segunda sección se compone de 13 preguntas cerradas, cuyo objetivo es evaluar el nivel de conformidad relativo al proceso

de calidad de software actual, en esta sección se utilizó la escala de Likert[1] y en adición se realizó una pregunta abierta con el objetivo de obtener posibles propuestas de mejora en el proceso.

Los resultados de la encuesta demostraron que el 61% del personal de desarrollo y de gestión de proyectos está de acuerdo con que las prácticas sugeridas por CMMI ayudarían a mejorar el proceso de calidad de software de la empresa, mientras que solo el 14% estuvo en desacuerdo.

Por otro lado, solo el 39% del personal está de acuerdo con que se sigue una política para la gestión de requerimientos, esto significa que un 61% está en desacuerdo o no conoce o utiliza esta política, de la misma manera solo el 50% entiende que se sigue una política para el aseguramiento de la calidad, mientras que el 50% restante está en desacuerdo o no conoce o utiliza esta política. Con relación a si la empresa cuenta con los recursos necesarios para llevar a cabo las actividades de aseguramiento de calidad, solo el 46% del personal estuvo de acuerdo, esta preocupación también se vio demostrada en las respuestas a la pregunta abierta en la que algunos colaboradores expresaron que la empresa no cuenta con los recursos necesarios para que estas actividades se lleven a cabo de manera óptima. De igual manera, solo el 47% de los encuestados estuvo de acuerdo en que las actividades de calidad son revisadas periódicamente, lo que indica que un 53% de los colaboradores está en desacuerdo o se mantiene al margen en cuanto a esta situación.

Ya que las políticas de gestión de requerimientos y calidad influyen de manera directa en el éxito de los proyectos y la satisfacción del cliente, queda demostrado que existe una necesidad de mejora en el proceso de calidad de software de la empresa.

Tomando estos resultados como referencia, se procedió a elaborar una propuesta del proceso de calidad de software que resuelve las inconformidades mostradas por el personal y observadas por medio de los métodos de investigación aplicados. Esta propuesta está basada en las mejores prácticas recomendadas por el modelo de mejora internacional y de renombre CMMI.

[1] La escala de Likert es una herramienta de medición que permite medir actitudes y conocer el grado de conformidad del encuestad. Se denomina así por Rensis Likert, quien publicó en 1932 un informe donde describía su uso.

INTRODUCCIÓN

Cuando se empezaron a construir los primeros productos de software, no se habían desarrollado los procesos necesarios para llevar a cabo esta tarea sin que ocurran problemas mayores, a esto se le llamó “La crisis de software”. A raíz de esta crisis, se creó la Ingeniería de Software como profesión, la cual se encargó de estructurar la manera en que se construían los productos, también ayudó a minimizar riesgos y a reducir costos. A partir de esto también se desarrolló lo que hoy conocemos como ciclo de vida de un software.

Con la implementación de la Ingeniería de Software, muchos errores fueron reducidos, pero, los incidentes seguían pasando porque los mismos desarrolladores eran los encargados de realizar pruebas a lo que construían. A lo largo de la historia se registraron eventos que se convirtieron en pérdidas millonarias, algunos de estos fueron cohetes que explotaron por errores de software, radiación excesiva por parte de máquinas de quimioterapia, entre otras catástrofes que resultaron en pérdidas de vida humana. Esas fueron algunas de las razones por la cual surgió un área dedicada al aseguramiento de la calidad de software.

La calidad de software hace énfasis en que procesos como la especificación de requerimientos, el análisis, el diseño y el producto final cumplan con las necesidades de los usuarios y que tengan un funcionamiento correcto.

No cabe duda de que el aseguramiento de la calidad de software se ha convertido en el proceso clave que se utiliza para evaluar si el producto desarrollado cumple con los estándares establecidos, de hecho, algunos autores se refieren a las pruebas de software (una de las actividades englobadas en el proceso de calidad) como la única actividad que se utiliza para asegurar la calidad de los productos de software.

La industria del software a lo largo de los años se ha desarrollado para que el ciclo de vida del software sea más completo. En este ciclo existe una parte de prueba, esta busca disminuir considerablemente la cantidad de defectos que puede tener un producto software, siendo esto a lo que se le llama calidad de software. Esta calidad debería estar presente en cada proyecto o empresa de software que quiera entregar un buen producto.

La empresa Wepsys S.R.L. tiene una vasta experiencia en el desarrollo de software, donde cuenta con varios equipos y procesos definidos de software. Los procesos de esta empresa se enfocan mucho en la parte de desarrollo, donde a través de los años han demostrado su capacidad para desplegar software de manera rápida y audaz. Sin embargo, esa velocidad ha dejado muestra de debilidades en los procesos de calidad de software a lo interno de lo empresa.

En vista de esta situación, se ha desarrollado este trabajo de grado, cuyo propósito es evaluar el proceso de calidad de software de la factoría de software Wepsys S.R.L. y proponer mejoras en base a los resultados. Para alcanzar este objetivo, este trabajo se basa en cuatro objetivos específicos que se han desarrollado en los siguientes capítulos:

- El primer capítulo titulado **"Marco teórico"**, abarca el objetivo número uno, que busca definir los conceptos que sustentan la investigación del análisis del estado actual del proceso de calidad de la factoría de software de Wepsys S.R.L.
- El segundo capítulo es la **"Metodología de investigación"**, engloba el objetivo número 2, que consiste es definir la metodología usada en la investigación, la cual sirvió para llevar a cabo el presente trabajo de grado.
- El tercer capítulo titulado **"Análisis detallado sobre el estado actual del proceso de calidad de software"**, abarca el tercer objetivo específico, el cual ofrece una descripción de sobre cómo se encuentra el estado actual del proceso de calidad de software de la empresa Wepsys S.R.L. En este capítulo se muestran datos estadísticos de cómo es la percepción de los colaboradores con respecto al proceso de calidad en la empresa.
- El cuarto y último capítulo, tiene como objetivo la presentación de **"Propuesta de mejoras al proceso de calidad de software"**. Dicha propuesta, consta de mejoras para el proceso en base a herramientas, técnicas y metodologías bien estandarizadas en el mundo de los procesos de calidad de software.

La novedad de este trabajo de investigación es que se utilizan las mejores prácticas recomendadas en el área clave de proceso PPQA del modelo internacionalmente reconocido CMMI. Este trabajo a su vez podrá ser utilizado por las autoridades pertinentes para realizar mejoras en el proceso de calidad de software, con el objetivo de mejorar la calidad del producto software entregado a los usuarios finales. Dicha mejora disminuirá considerablemente la cantidad de defectos encontrados en ambientes

productivos y, por ende, se bajarán los costos que representan reparar un defecto, esto puesto que dicho defecto se encontraría en etapas tempranas del desarrollo. De esta manera, el ciclo de vida del software en Wepsy S.R.L. será más eficiente y de mayor calidad.

CAPÍTULO I: MARCO TEÓRICO

Introducción

La calidad de software es el norte al que todos los proyectos de software deberían estar orientados, ya que todas las prácticas que se han desarrollado a lo largo de la historia, incluyendo a la Ingeniería de Software y los modelos de mejora como el Modelo de Madurez de Capacidad Integrado (CMMI), tienen como propósito que el software sea construido utilizando las buenas prácticas y que las necesidades de los clientes y/o usuarios queden satisfechas.

En este capítulo, se definen los conceptos principales que permiten comprender por qué la calidad del software es una práctica importante dentro de las empresas que en específico se dedican a comercializar software y como todos los avances que se han realizado en el ámbito del desarrollo de software han sido gracias a la búsqueda constante de aumentar la calidad.

1.1 Ingeniería de Software

El Instituto de Ingeniería Eléctrica y Electrónica, mejor conocido como IEEE, por sus siglas en inglés, define el **software** como "Conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación".

Por otro lado, las **ingenierías** son el conjunto de técnicas y conocimientos que junto a la invención aprovechan la materia prima de un área para la actividad industrial.

Estas definiciones indican el camino para deducir qué es la **ingeniería de software**, la cual trata del establecimiento de métodos para el diseño y la construcción de software y ayuda a documentar, operar y mantener el desarrollo de estos. Autores como Somerville (2016), la definen como "una disciplina de la ingeniería que se ocupa de todos los aspectos de la producción de un software, desde su concepción hasta su operación y mantenimiento".

Otras definiciones aceptadas proponen que la Ingeniería de software abarca un proceso, una colección de métodos (prácticas) y una variedad de herramientas que permiten a los profesionales construir software de computadora de alta calidad (Pressman & Maxim, 2015).

1.1.1 Antecedentes

Para hablar de la historia de la ingeniería de software, tenemos que remontarnos a 1964 cuando IBM combinó su sistema de computadora IBM S/360 con características tanto científicas como de negocios, este conjunto de máquinas hizo que las personas trataran de desarrollar sistemas de software más grandes y complejos generando así la necesidad de tener una disciplina que maneje el desarrollo de software.

En 1969, en la conferencia de NATO sobre software engineering fue la primera vez que se utilizó el término de ingeniería de software, el cual fue acreditado a Fritz Bauer. Personajes como Peter Naur y Edsger Dijkstra también realizaron grandes aportes para que la ingeniería de software crezca en esos años.

La primera conferencia internacional sobre ingeniería de software fue en 1973 por el IEEE. En los años siguientes Barry Boehm realizó un famoso documento titulado Software Engineering, enmarcando el objetivo de la ingeniería de software. Ya en el 1975 Frederick Brooks, quien fue director del proyecto de desarrollo del sistema operativo de IBM 360 por un periodo de 10 años escribió un libro muy famoso en el área de ingeniería de software el cual se titula “The Mythical Man-Month” donde se mostraron varios problemas con respecto al desarrollo de software en un ambiente donde trabajan numerosas cantidades de personas.

Durante los años 1960 y 1970, hubo una gran crisis con respecto al software, una crisis con síntomas que incluso duraron hasta la actualidad. Uno de estos síntomas era que el software estaba incrementando mucho en costo y la calidad era pobre.

En la década del 1970 se desarrollaron distintas técnicas y métodos de ingeniería de software, algunas de estas técnicas fueron la programación estructurada, la programación orientada a objetos y distintos estándares que son la base de la ingeniería de software que conocemos hoy en día (Sommerville, 2016).

1.1.2 Ciclo de vida del software

Es un esquema o esqueleto que se aplica a un producto de software. A través de los años se han desarrollado diferentes modelos a seguir para establecer un proceso de software efectivo. Cada uno de estos modelos utilizan unas actividades diferentes durante el proceso para poder llegar así al producto final. Entre las actividades que deben realizarse durante el ciclo de vida de un proyecto de software se encuentra la planificación, implementación, pruebas, documentación, despliegue y mantenimiento.

La **planificación** consiste en adquirir los requerimientos que el cliente tiene en su cabeza para analizarlos y transformarlos en funciones y características que el software debería cumplir.

Por otra parte**, la implementación** es donde los desarrolladores escriben el código necesario para cumplir los requerimientos y demandas del cliente.

Las **pruebas** son un conjunto de actividades que cubren el ciclo de vida del software entero y que tiene como propósito localizar defectos (Chopra, 2018).

La **documentación** ayuda a facilitar la mejora y mantenimiento del software durante su desarrollo y puesta en producción. Esta documentación puede incluir tanto diseño interno del software, como información de ambiente de este o incluso documentación de interfaces públicas como privadas.

El **despliegue** comienza cuando el código está listo para pasar a un ambiente de producción, pero no sin antes pasar por todas las pruebas concernientes a su liberación.

El mantenimiento o mejora de software se centra en resolver los nuevos problemas que han sido desplegados, aquí se pueden agregar incluso más piezas de código que el diseño original todo para solucionar el problema o incrementar las funciones del software.

1.1.3 Metodologías de desarrollo de software y su importancia

Para desarrollar un producto de software es necesario seguir un conjunto de pasos estructurados que permitan convertir los requisitos de los clientes en el producto deseado, que se realice a tiempo y que sea funcional, es a este proceso completo que se le conoce como metodologías o modelos de desarrollo de software.

Estas metodologías surgieron para dar solución a la llamada crisis del software, la cual se caracterizó de softwares poco fiables y de alto costo, y siguieron evolucionando hasta convertirse en lo que conocemos en la actualidad.

A las primeras metodologías se les llamó metodologías tradicionales, pero con la llegada del internet surgieron proyectos cuyos requisitos eran cambiantes y se contaba con un corto plazo, esto hizo que los modelos tradicionales no cumplieran las necesidades de estos, lo cual abrió paso a lo que conocemos como metodologías ágiles. (Zumba & Cecibel, 2018).

Metodologías tradicionales

Durante la década del 1960, lo que se utilizaba era el método de "programa y arregla", pero, no era un método efectivo, por lo que Winston Royce en el 1970 propuso la metodología conocida como cascada.

Este modelo se basa en una serie de fases que tienen definida una serie de actividades y entregables que solo se pueden completar antes de que la fase siguiente comience. Luego de años de utilizar el modelo cascada e incremental Barry Boehm expuso los problemas de estos métodos y propuso el modelo en espiral. El **modelo en espiral** integra lo mejor de los anteriores y trata de iterar las fases.

A pesar de que el modelo en espiral tiene un enfoque evolutivo, otras metodologías tradicionales surgieron, una de estas es el proceso unificado, que en su máximo esplendor es descrita como una metodología iterativa e incremental que hace énfasis en la documentación y en el uso de artefactos de software como los distintos diagramas UML, los casos de uso, entre otros.

Metodologías ágiles

Los métodos ágiles combinan los modelos iterativos e incrementales con la practicidad de liberar el producto por entregables, en lugar de un producto completo una sola vez. Fueron desarrollados con la mentalidad de hacer el proceso de desarrollo más rápido o como su nombre describe, más ágil y de transformar el desarrollo de software en un proceso totalmente evolutivo orientado a la calidad. Algunas metodologías ágiles son Extreme Programming (XP), Scrum y Kanban.

1.1.4 Desarrollo de software en la actualidad

Según Dooley (2017), El Desarrollo de software es un proceso que forma parte de la ingeniería de software y se encarga de tomar un grupo de requerimientos de un usuario, analizarlos, diseñar una solución e implementarla.

La práctica del desarrollo de software se había puesto en marcha mucho antes de que surgiera la ingeniería de software. En 1948, Tom Kilburn fue el primero que escribió un pedazo de software.

Fue en 1970 y 1980 con la llegada de las computadoras personales que el desarrollo de software alcanzó su pico. Luego de la inestabilidad del área en las décadas anteriormente mencionadas, surge la creación de la ingeniería de software, gracias a la cual hoy en día existen prácticas definidas y metodologías estables, que hicieron de esta práctica uno de los motores de la sociedad y la economía actual.

En la actualidad, el desarrollo de software es un campo laboral estable, aunque frecuentemente se crean o se mejoran lenguajes de programación, arquitecturas, herramientas, etc.

De acuerdo con Evans Data Corporation (2019), en el mundo había 23.9 millones de desarrolladores de software a mayo del 2019 y se estima que para el 2024 habrán 28.7 millones de personas que se dedicarán a esta profesión.

Estados Unidos es el país que tiene la delantera con 4.2 millones, pero se proyecta que para 2024, India tome el puesto que hoy ostenta Estados Unidos.

La región del pacífico de Asia es la que cuenta con la mayor tasa de crecimiento, mientras que Latinoamérica se encuentra en segundo lugar (ver figura 1.1).

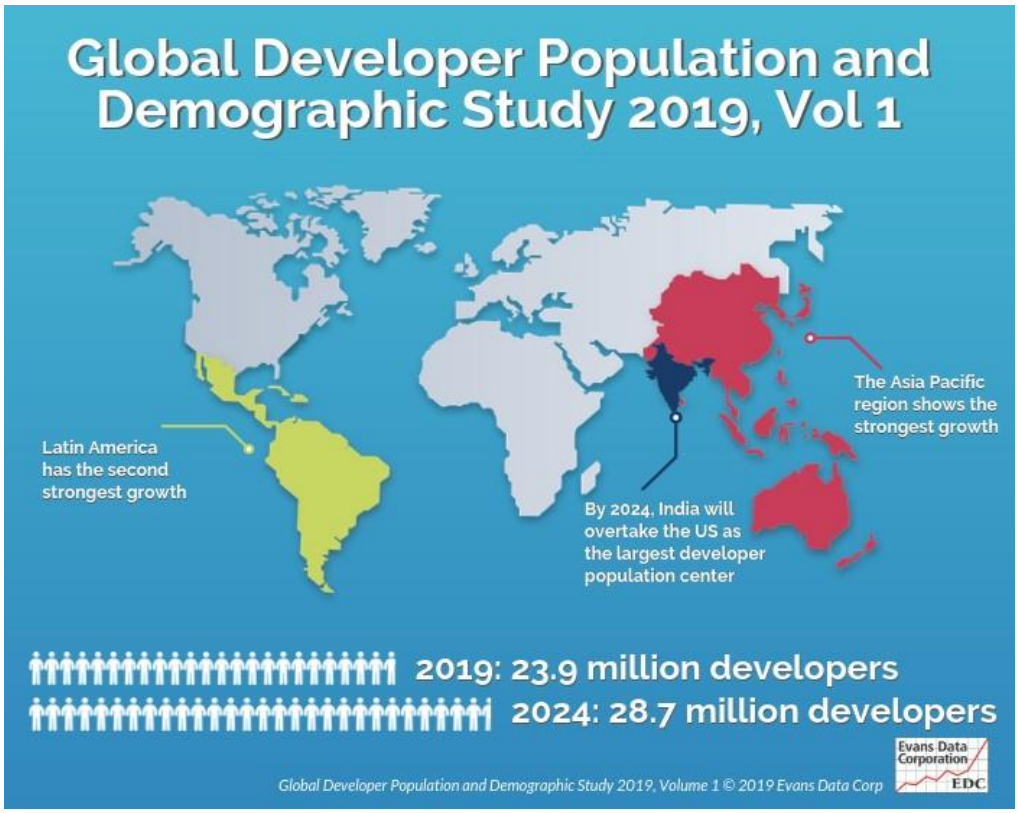


Figura 1.1 - Estudio demográfico de la población mundial de desarrolladores de software 2019.

1.1.4.1 Desarrollo de software en República Dominicana

En República Dominicana la industria del software se puede describir como incipiente o poco desarrollada. Este sector surgió con un enfoque de construir sistemas administrativos y financieros que ayuden a volver más eficiente a los procesos de esta índole.

El 16% de sistemas que se realizan en el país son de cara al sector público, un 15% para el sector bancario y un 13% para el manejo de inventarios. Esto indica que tan poco desarrollada está el área de software, dónde en otros países ya se está trabajando en conjunto con la Inteligencia artificial para dar solución a problemas de carácter social.

Según un análisis presentado por el Observatorio MiPymes (2020), auspiciado por el Ministerio de Industria, Comercio y MiPymes, el programa República Digital y el Instituto Tecnológico de Santo Domingo (INTEC), en el país, para el año 2017 había un total de 2,253 MiPymes TIC, del cual el 58.4% están concentradas en el Distrito Nacional, mientras que un 15.8% en Santo Domingo y un 9.5% en Santiago.

Sin embargo, de acuerdo con el Registro Nacional de Establecimientos (2014), es una meta para el país trabajar para el desarrollo de este sector debido a que es una actividad de alto impacto económico, con una alta tasa de innovación, competitividad y productividad, convirtiéndose en una fuente de reducción de pobreza. Este sector ha creado aproximadamente 39,000 empleos.

De todas MiPymes TIC que existen en República Dominicana, la mayoría de estas no cuentan con un proceso de calidad separado del equipo de desarrollo de software, lo cual incrementa exponencialmente, el riesgo a comercializar productos defectuosos, a invertir más dinero del presupuestado y a tardar más para operar en un ambiente de producción.

1.1.5 Calidad

Según Mitra (2016), la calidad se define como la aptitud de un producto o servicio para cumplir las expectativas de acuerdo con los requerimientos del cliente.

Una de las palabras clave de la definición anterior es la palabra "cliente", por tanto, la calidad no es objetiva, sino que implica un sin número de variables como expectativas, distintos grupos de usuarios, etc. La calidad tiene características que se pueden clasificar en dos tipos:

Variables: Son características que se pueden medir y se expresan en una escala numérica.

Atributos: Son características de la calidad que no pueden ser medidas en una escala numérica.

Dos conceptos relativos a la calidad que son muy importantes son:

Control de calidad: Se puede definir como un sistema de mecanismos que se realizan o utilizan para detectar errores.

Aseguramiento de la calidad: Conjunto de actividades planificadas y aplicadas para que los requisitos de un producto o servicio sean satisfechos.

1.2 La calidad del software

La calidad de software incluye distintos atributos y puede ser definida y percibida de diferentes maneras de acuerdo con los distintos roles y responsabilidades, dicho esto, podemos decir que un software tiene calidad cuando cumple con los requerimientos que fueron levantados y cuando satisface las necesidades de los clientes y/o usuarios.

La actividad que se utiliza para determinar si un software tiene calidad o no, son las pruebas de software. Las pruebas de software se definen como el proceso de ejecutar un programa o sistema con el propósito de encontrar algún defecto.

1.2.1 Importancia

La calidad es el principal enfoque de la construcción de cualquier sistema de software. Asegurar la calidad del software es importante por las siguientes razones:

- Se detectan y/o previenen defectos que pueden afectar a la organización económica, social y ambientalmente o que incluso pueden afectar la salud de las personas.
- Los costes económicos se reducen considerablemente.
- Un software de calidad ayuda a la empresa a tener una buena reputación.
- Un software de calidad ayuda a construir confianza y a ampliar la cartera de clientes, gracias a futuras recomendaciones.

1.2.2 Proceso de aseguramiento de calidad de software

Para comenzar a hablar del aseguramiento de la calidad de software, tenemos que definirlo, este proceso es un conjunto de métodos, herramientas y técnicas que permiten gestionar la calidad en el desarrollo de un producto de software (Carrizo & Alfaro, 2018). La ISO 17a lo define como la aplicación sistemática de conocimientos, métodos y experiencia científicos y tecnológicos para el diseño, implementación, prueba y documentación de software (Claude Y. Laporte, 2018).

Este conjunto de métodos es fundamental para el desarrollo, sin embargo, son pocas las empresas de software que lo implementan debido a diversos factores como los son la falta de presupuesto, falta de personal o la falta de capacidad para adaptarse a estándares más complejos. Para la IEEE, el aseguramiento de la calidad se define como un conjunto de actividades que evalúan la idoneidad de los procesos de software para proporcionar

evidencia que establezca la confianza de que los procesos de software son apropiados y producen productos de software de calidad adecuada para los fines previstos.

El proceso de aseguramiento de la calidad de software se vale de herramientas, esencia y métricas para así asegurar la calidad del software. Las herramientas de aseguramiento de la calidad tienen como objetivo controlar la calidad de un proyecto de software, esto con la finalidad de que el proyecto tenga la calidad esperada. La parte de la esencia, que tiene como objetivo ayudar al equipo de trabajo a entender lo que es el concepto de calidad, que no solo se basa en tareas o actividades, sino que también en cómo el equipo se desenvuelve. Por otra parte, están las métricas que son utilizadas para medir resultados, pero además estas sirven para mejorar los procesos internos (Carrizo & Alfaro, 2018).

Para llevar a cabo el aseguramiento de la calidad del software de manera adecuada, es importante que se evalúe y se divulguen los datos sobre el proceso de ingeniería de software. Estas estadísticas junto al aseguramiento de calidad de software ayudan mejorar la calidad del producto y del proceso de software.

1.2.3 Las pruebas de software

Los sistemas de software hoy en día son parte importante e integral en nuestras actividades diarias; un ejemplo de ello son los celulares inteligentes, las tabletas, los relojes inteligentes o los Smart TV. Estos dispositivos o aplicaciones son creadas e implementadas por seres humanos y por ende en cualquiera de sus etapas de desarrollo se puede generar o presentar una equivocación que puede llevar a un defecto. Si no se ha identificado ese defecto y la o las aplicaciones se ejecutan, hay un alto riesgo de que la aplicación no haga lo que debería hacer o el objeto para lo cual fue creada, es decir se genera un fallo o desperfecto (Pardo, Hurtado, & Collazos, 2018).

En 1957 se conoce la prueba el Debugging, y Dijkstra en 1970 presenta una afirmación: “La prueba de software puede ser usada para mostrar la presencia de bugs, pero nunca la ausencia” (Dijkstra). Una prueba de software es una verificación que se realiza contra el comportamiento de un programa contra un comportamiento esperado, usando una cantidad de casos finitos de pruebas, que son seleccionados de manera adecuada, esto lo aclara (IEEE).

Muchas veces las pruebas de software son realizadas a diferentes niveles durante el proceso de desarrollo y el proceso de mantenimiento. Estos niveles se pueden diferenciar por el objeto de prueba, el cual se le llama blanco de pruebas u objetivo.

Entre los niveles de prueba se encuentran:

1. La prueba de objetivo general

Esta se basa en probar un módulo simple, o un conjunto de módulos relacionados con un propósito o comportamiento. Entre las subpruebas que se encuentran dentro de este nivel, están:

a) Unit Testing

El unit testing se contempla como una prueba que verifica una funcionalidad aislada de un software, generalmente un pedazo de software por separado.

b) Integration Testing

El integration testing es un proceso que verifica la interacción entre componentes de software. Las pruebas de integración por lo general es una actividad que se realiza durante cada etapa del desarrollo de software durante la cual los desarrolladores obtienen las perspectivas del nivel que están integrando.

c) System Testing

La prueba de Sistema se ocupa de probar el comportamiento de todo el sistema como tal, básicamente prueba la efectividad en unidad del software. Este tipo de pruebas es ideal para probar requisitos de calidad de software tanto funcionales como no funcionales.

2. Prueba de objetivos específicos

Este nivel de pruebas, se realizan teniendo en cuenta objetivos específicos, estos objetivos se establecen de forma más o menos explícita y con distintos grados de precisión. Entre el conjunto de subpruebas se encuentran:

a) Pruebas de aceptación

Determinan si un sistema cumple con los criterios de aceptación, este generalmente verifica los comportamientos del sistema contra los requisitos que el cliente pidió. Este tipo de prueba no debe incluir desarrolladores del sistema.

b) Pruebas de instalación

Por lo general, después que se completan las pruebas de aceptación y de sistema, el software pasa por una instalación por el ambiente de destino. Las pruebas de instalación

son en esencia pruebas de sistemas realizadas en el ambiente operativo de las configuraciones del hardware y restricciones operativas.

c) Pruebas alpha y beta

Antes de que el producto de software se publique, muchas veces los equipos optan por facilitar el software a grupos de usuario pequeños (pruebas alfa) y grupos de usuarios más representativos (pruebas beta), esto con la finalidad que los usuarios reporten problemas con el producto que no salieron a relucir en las pruebas controladas.

d) Logro y evaluación de confiabilidad

Las pruebas mejoran la confiabilidad al identificar y corregir fallas. Además, las medidas estadísticas de confiabilidad pueden derivarse generando aleatoriamente casos de prueba de acuerdo con el perfil operacional del software.

e) Pruebas de regresión

Son pruebas selectivas de un sistema o componente de software para verificar que las modificaciones realizadas no hayan afectado o causado cambios no deseados y que el sistema o componentes de software sigan cumpliendo con los requisitos específicos.

f) Pruebas de Performance

La prueba de performance verifica que el software cumple con los requisitos de rendimientos especificados y evalúa las características de rendimiento para instancia, capacidad y tiempo de respuesta.

g) Pruebas de seguridad

La prueba de seguridad se centra en la verificación de que el software está protegido contra ataques externos. Estas pruebas se enfocan en verificar la confidencialidad, integridad y disponibilidad de los sistemas y sus datos.

h) Pruebas de Stress

Las pruebas son pruebas las cuales llevan al sistema al máximo, tanto en carga de diseño, así como rendimiento, con el objetivo de probar mecanismos de defensa en sistemas críticos

i) Pruebas back to back

La norma IEEE / ISO / IEC 24765 define las pruebas consecutivas como "prueba en las que dos o más las variantes de un programa se ejecutan con el mismo entradas, las salidas se comparan y los errores se analizados en caso de discrepancias ".

j) Pruebas de recuperación

Están destinadas a verificar que el software reinicia sus capacidades después de un bloqueo del sistema u otro desastre.

k) Pruebas de Interfaz

Estas pruebas buscan verificar si los componentes se interconectan correctamente para proporcionar el correcto intercambio de los datos e informaciones de control. Estas pruebas simulan el uso de un API mediante aplicaciones de usuario final. Esto implica la generación de parámetros a las llamadas API, las configuraciones de condiciones ambientales externas y la definición de datos internos que afectan a la API.

l) Pruebas de configuración

Verifican el software bajo diferentes configuraciones.

m) Pruebas de usabilidad y de interacción humana

La prueba de interacción es evaluar qué tan fácil es para que los usuarios finales aprendan y utilicen el software. En general, puede implicar probar las funciones de software que respaldan las tareas del usuario, documentación que ayuda a los usuarios y la capacidad del sistema para recuperarse de errores de usuario.

1.2.4 Actividades del ciclo de pruebas

La calidad de software está ligada a la validación y verificación de los requerimientos. Para poder alcanzar la calidad deseada, las actividades del ciclo de vida del software son fundamentales, estas como se ven en la figura 1.2, forman un ciclo perfecto el cual se define como una serie de actividades que se realizan para realizar pruebas de software. Por lo tanto, el Ciclo de vida de las pruebas de software se refiere a un proceso de prueba con pasos específicos.

Estas pruebas se realizan con los siguientes pasos en un orden específico para garantizar que el software cumpla con los objetivos de calidad. En el proceso ciclo de vida de pruebas de software, por lo tanto, llevamos a cabo cada actividad de forma planificada y sistemática. Cada fase tiene sus propios objetivos y resultados. Las fases del Ciclo de vida de pruebas de software son diferentes para cada organización; Sin embargo, la base sigue siendo la misma.

Figura 1.2 - Actividades del ciclo de pruebas

Planeación de Pruebas

Es la etapa en donde se ejecutan las primeras actividades correspondientes al proceso de pruebas y tiene como resultado un entregable denominado plan de pruebas el cual debe estar conformado en cuando menos por aspectos tales como:

Alcance de la prueba: determina qué funcionalidades del producto y/o software serán probadas durante el transcurso de la prueba. Este listado de funcionalidades a probar se extrae con base a un análisis de riesgos realizado de manera previa, que tienen en cuenta variables tales como el impacto que podría ocasionar la falla de una funcionalidad y la probabilidad de falla de una funcionalidad. Producto de este análisis, se cuenta con información adicional que permite determinar además del alcance detallado del proceso de pruebas, la prioridad con la que las funcionalidades deben probarse y la profundidad de las pruebas.

Tipos de Prueba: en este punto se debe determinar qué tipos de pruebas requeriría el producto. No todos los productos de software requieren la aplicación de todos los tipos de pruebas que existen, por esta razón, es estrictamente necesario que el líder de pruebas se plantee preguntas que le permitan determinar qué tipos de prueba son aplicables al proyecto en evaluación. Los posibles tipos de prueba a aplicar son: pruebas de stress,

pruebas de rendimiento, pruebas de carga, pruebas funcionales, pruebas de usabilidad, pruebas de regresión, entre otros.

Estrategia de Pruebas: teniendo en cuenta que no es viable probar con base a todas las posibles combinaciones de datos, es necesario determinar a través de un análisis de riesgos sobre qué funcionalidades debemos centrar nuestra atención. Adicionalmente, una buena estrategia de pruebas debe indicar los niveles de pruebas (ciclos) que aplicaremos y la intensidad o profundidad a aplicar para cada nivel de pruebas definido. En este punto también es importante definir los criterios de entrada y salida para cada ciclo de pruebas a ejecutar.

Criterios de Salida: entre las partes involucradas en el proceso, se define de manera formal, en qué condiciones se puede considerar que una actividad de pruebas fue finalizada. Los criterios de salida se deben definir para cada nivel de pruebas a ejecutar. Algunos ejemplos de criterios de salida que pueden ser utilizados son: porcentaje de funcionalidades de alto riesgo probadas con éxito, número defectos críticos y/o mayores aceptados, etc.

Otros aspectos: tal y como se realiza en cualquier plan de proyecto, se debe incluir una estimación de tiempos, los roles y/o recursos que harán parte del proceso, la preparación del entorno de pruebas, cronograma base, etc.

Diseño de Pruebas: una vez elaborado y aprobado el plan de pruebas, el equipo de trabajo debe iniciar el análisis de toda la documentación existente con respecto al sistema, con el objeto de iniciar el diseño de los casos de prueba. Los entregables claves para iniciar este diseño pueden ser: casos de uso, historias de usuario, arquitectura del sistema, diseños, manuales de usuario (si existen), manuales técnicos (si existen). El diseño de los casos debe considerar la elaboración de casos positivos y negativos. Los casos de prueba negativos permiten validar cómo se comporta el sistema ante situaciones atípicas y permite verificar la robustez del sistema, atributo que constituye uno de los requerimientos no funcionales indispensable para cualquier software. Por último, es necesario definir cuáles son los datos de prueba necesarios para la ejecución de los casos de prueba diseñados.

Implementación y Ejecución de Pruebas: la ejecución de pruebas debe iniciar con la creación de los datos de prueba necesarios para ejecutar los casos de prueba diseñados.

La ejecución de estos casos puede realizarse de manera manual o automatizada; en cualquiera de los casos, cuando se detecte un fallo en el sistema, este debe ser documentado y registrado en una herramienta que permita gestionar los defectos (Bug Tracker). Una vez el defecto ha sido corregido por la firma desarrolladora en su respectivo proceso de depuración, es necesario realizar un re-test que permita confirmar que el defecto fue solucionado de manera exitosa. Por último, es indispensable ejecutar un ciclo de regresión que nos permita garantizar, que los defectos corregidos en el proceso de depuración de la firma no hayan desencadenado otros tipos de defectos en el sistema.

Evaluación de Criterios de Salida: los criterios de salida son necesarios para determinar si es posible dar por finalizado un ciclo de pruebas. Para esto, es conveniente definir una serie de métricas que permitirán al finalizar un proceso de pruebas, comparar los resultados obtenidos contra las métricas definidas, sí los resultados obtenidos no superan las métricas definidas, no es posible continuar con el siguiente ciclo de pruebas.

Existen varios tipos de criterios de salida dentro de los cuales se pueden mencionar: cubrimiento de funcionalidades en general, cubrimiento de funcionalidades críticas para el sistema, Número de defectos críticos y mayores detectados, etc. También es importante aclarar que el proceso de pruebas puede ser suspendido y/o paralizado, debido entre otros, a aspectos relacionados con el presupuesto y la calidad mínima del sistema requerida para el inicio formal de pruebas.

Cierre del proceso: durante este periodo de cierre el cual históricamente se ha comprobado que se le destina muy poco tiempo en la planeación, se deben cerrar las incidencias reportadas, se debe verificar si los entregables planeados han sido entregados y aprobados, se deben finalizar y aprobar los documentos de soporte de prueba, analizar las lecciones aprendidas para aplicar en futuros proyectos, etc.

1.3 Modelo de Madurez de Capacidad Integrado (CMMI)

Modelo de Madurez de Capacidad Integrado (**CMMI**) es un programa de evaluación y capacitación de mejora a nivel de proceso. Administrado por el Instituto CMMI, una subsidiaria de ISACA, fue desarrollado en la Universidad Carnegie Mellon (CMU). Es requerido por muchos contratos del gobierno de EE. UU., Especialmente en el desarrollo de software. CMU afirma que CMMI se puede utilizar para guiar la mejora de procesos en un proyecto, división o toda una organización. CMMI define los siguientes niveles de

madurez para los procesos: inicial, administrado, definido, administrado cuantitativamente y optimizado.

Originalmente, CMMI aborda tres áreas de interés:

1) Desarrollo de productos y servicios: CMMI para el desarrollo (CMMI-DEV).
2) Establecimiento, gestión de servicios: CMMI para servicios (CMMI-SVC).
3) Adquisición de productos y servicios: CMMI for Acquisition (CMMI-ACQ).

En la versión 2.0, estas tres áreas (que anteriormente tenían un modelo separado cada una) se fusionaron en un solo modelo.

CMMI fue desarrollado por un grupo de la industria, el gobierno y el Instituto de Ingeniería de Software (SEI) en CMU. Los modelos CMMI brindan orientación para desarrollar o mejorar los procesos que cumplen con los objetivos comerciales de una organización. Un modelo CMMI también se puede utilizar como marco para evaluar la madurez del proceso de la organización. En enero de 2013, todo el conjunto de productos CMMI se transfirió del SEI al CMMI Institute, una organización recién creada en Carnegie Mellon.

1.3.1 Antecedentes

CMMI fue desarrollado por el proyecto CMMI, cuyo objetivo era mejorar la usabilidad de los modelos de madurez mediante la integración de muchos modelos diferentes en un marco. El proyecto estuvo integrado por miembros de la industria, el gobierno y el Carnegie Mellon Software Engineering Institute (SEI). Los patrocinadores principales incluyeron la Oficina del Secretario de Defensa (OSD) y la Asociación Industrial de Defensa Nacional.

CMMI es el sucesor del modelo de madurez de capacidad (CMM) o Software CMM. El CMM se desarrolló desde 1987 hasta 1997. En 2002, se lanzó la versión 1.1, la versión 1.2 siguió en agosto de 2006 y la versión 1.3 en noviembre de 2010. Algunos cambios importantes en CMMI V1.3 son el soporte del desarrollo de software ágil, mejoras en las prácticas de alta madurez y alineación de la representación (por etapas y continua).

Según el Instituto de Ingeniería de Software (SEI, 2008), CMMI ayuda a "integrar funciones organizativas tradicionalmente separadas, establecer objetivos y prioridades de mejora de procesos, proporcionar orientación para los procesos de calidad y proporcionar un punto de referencia para evaluar los procesos actuales".

En marzo de 2016, el Instituto CMMI fue adquirido por ISACA.

1.3.2 Organización y composición del Modelo de Madurez de Capacidad Integrado (CMMI)

1.3.2.1 Organización

En la versión 1.3 de CMMI, este modelo de mejora se organiza en constelaciones. Una constelación es definida como un conjunto o colección de componentes CMMI que se utilizan para el desarrollo y construcción de modelos y materiales de evaluación y formación en un área de interés.

Las constelaciones de CMMI en su versión 1.3 son:

Desarrollo (CMMI-DEV): es la guía que se utiliza para medir, monitorear y administrar todo en cuanto al proceso de desarrollo se refiere, también se encarga del mantenimiento de productos y servicios.

Adquisición (CMMI-ACQ): es la guía que se encarga de mejorar el proceso de adquisición de productos y servicios.

Servicios (CMMI-SVC): es la guía destinada para la guía de proporción de servicios internos en una organización y a clientes externos.

Estas constelaciones fueron unificadas en la versión 2.0, publicada en el 2018, la cual es la versión más reciente del Modelo de Madurez de Capacidad Integrado.

1.3.2.2 Composición del modelo

Según el Software Engineering Institute (2010), los componentes del modelo pueden ser agrupado en las siguientes categorías:

Componentes requeridos: son metas y objetivos genéricos que describen qué se debe realizar para satisfacer un área de negocio en calidad de organización.

Componentes esperados: estos plasman lo que le es posible hacer a una organización para lograr un componente requerido. Estos componentes son utilizados para conducir a los que implementan mejoras o realizan evaluaciones.

Componentes informativos: son aquellos que suministran detalles que sirven de ayuda a las organizaciones a empezar a pensar en la manera de aproximarse a los componentes requeridos y esperados.

1.3.3 Niveles del Modelo de Madurez de Capacidad Integrado (CMMI)

Como el nombre del modelo lo explica, el modelo CMMI se basa en capacidades y en madurez, por tanto, existen niveles para estas dos características (Chaudhary & Abhishek, 2017).

1.3.3.1 Niveles de Capacidad

Estos niveles se encargan de asegurar que la organización mejore las áreas de procesos individualmente. Existen cuatro niveles de capacidad:

Nivel 0 – Incompleto: esto significa que los objetivos específicos no han sido satisfechos para esa área de proceso y que no existen objetivos generales.

Nivel 1 – Realizado: esto significa que los objetivos específicos del área de proceso fueron satisfechos, pero se deben realizar mejoras al proceso y estas deben ser institucionalizadas y mantenidas o se pueden perder con el tiempo.

Nivel 2 – Gestionado: esto significa que el proceso fue planeado y ejecutado y que las prácticas a diferencia del nivel anterior se mantendrán durante tiempos cruciales.

Nivel 3 – Definido: esto significa que los procesos han sido adaptados en la organización.

1.3.3.2 Niveles de Madurez

Estos niveles se componen de prácticas específicas y genéricas relacionadas para un grupo de áreas de proceso que tienen el propósito de mejorar el rendimiento general de la organización. Estos niveles son:

Nivel 1 – Inicial: en este nivel los procesos son hechos para cumplir ciertos propósitos en específico y el éxito se basa en ciertas personas de la organización. En este nivel, los productos y servicios funcionan bien la mayoría del tiempo, pero el tiempo y el presupuesto siempre se ven afectados.

Nivel 2 – Gestionado: significa que los proyectos se apegan a los procesos definidos de la organización.

Nivel 3 – Definido: trata acerca de tener los procesos de la organización están bien definidos, documentados y entendidos.

Nivel 4 – Gestionado cuantitativamente: significa que la organización establece objetivos cuantitativos para la calidad de los procesos y se utilizan como criterios a considerar en la gestión de proyectos.

Nivel 5 – En optimización: la organización se enfoca en la mejora continua de procesos utilizando la comprensión cuantitativa de los objetivos del negocio.

1.3.4 Áreas de procesos del Modelo de Madurez de Capacidad Integrado (CMMI)

Un área de proceso es un grupo de prácticas que se relacionan en una zona, cuando estas áreas se implementan de forma conjunta cumplen o satisfacen un conjunto de objetivos importantes para mejorar el área. Estas pueden ser representadas en dos maneras:

Representación continua: el esfuerzo se enfoca en los conjuntos de áreas de procesos interrelacionadas según el nivel de capacidad. Con el objetivo de dar soporte a los que usan este tipo de representación, las áreas de procesos se categorizan en cuatro: Gestión de Procesos, Gestión de proyectos, Ingeniería y Soporte (Ver figura 1.3).

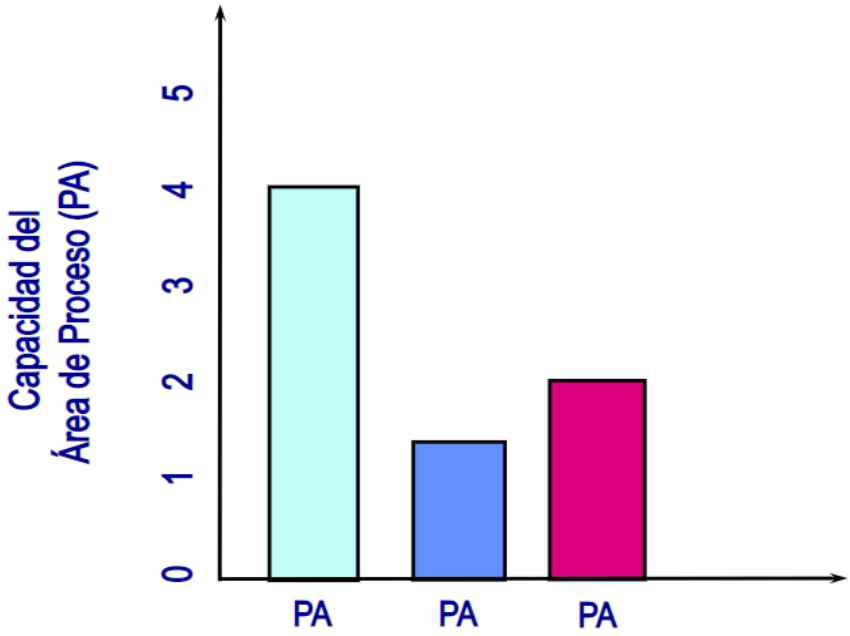


Figura 1.3 – Representación continua de las áreas de proceso.

Las áreas de proceso también pueden ser representadas por etapas, en este caso, se agrupan según el nivel de madurez (Maturity Level) al que pertenecen (Ver figura 1.4).

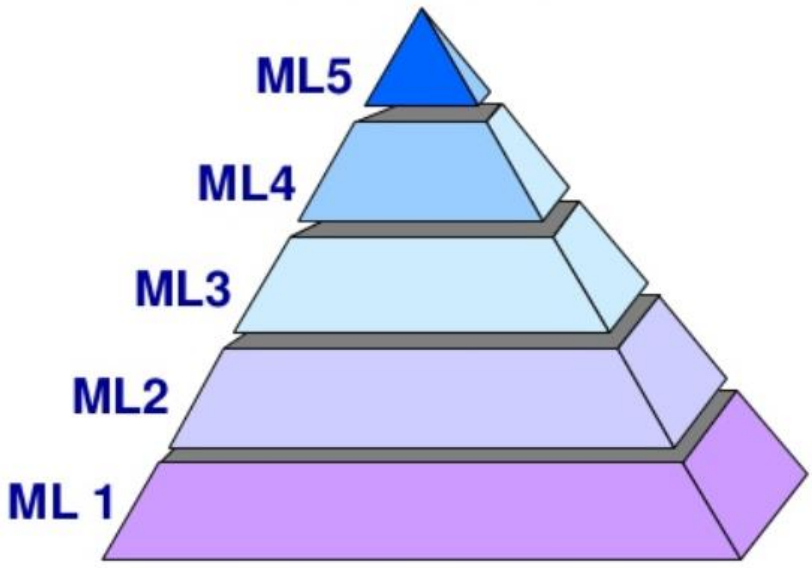


Figura 1.4 – Representación por etapas de las áreas de proceso.

Ya que se conoce como las áreas pueden ser representadas, en la tabla 1.1 se muestra una lista de las áreas de proceso de CMMI DEV versión 1.3.

Área de proceso	Categoría	Niveles de Madurez
Análisis Causal y Resolución (CAR)	Soporte	5
Gestión de Configuración (CM)	Soporte	2
Análisis de Decisiones y Resolución (DAR)	Soporte	3
Gestión Integrada del Proyecto (IPM)	Gestión de proyectos	3
Medición y Análisis (MA)	Soporte	2
Definición de Procesos de la Organización (OPD)	Gestión de procesos	3
Enfoque en Procesos de la Organización (OPF)	Gestión de procesos	3
Gestión del Rendimiento de la Organización (OPM)	Gestión de procesos	5
Rendimiento de Procesos de la Organización (OPP)	Gestión de procesos	4
Formación en la Organización (OT)	Gestión de procesos	3
Integración del Producto (PI)	Ingeniería	3
Monitorización y Control del Proyecto (PMC)	Gestión de proyectos	2
Planificación del Proyecto (PP)	Gestión de proyectos	2
Aseguramiento de la Calidad del Proceso y del Producto (PPQA)	Soporte	2
Gestión Cuantitativa del Proyecto (QPM)	Gestión de proyectos	4
Desarrollo de Requisitos (RD)	Ingeniería	3
Gestión de Requisitos (REQM)	Gestión de proyectos	2
Gestión de Riesgos (RSKM)	Gestión de proyectos	3
Gestión de Acuerdos con Proveedores (SAM)	Gestión de proyectos	2

Solución Técnica (TS)	Ingeniería	3
Validación (VAL)	Ingeniería	3
Verificación (VER)	Ingeniería	3

Tabla 1.1 – Áreas de proceso

Resumen del capítulo I

La ingeniería de software es un conjunto de métodos y prácticas desarrolladas con el propósito de aumentar la calidad de los productos de software y sin dudas todo lo que surgió a partir de la misma, tiene el mismo objetivo.

En este capítulo, se pudo visualizar qué es la ingeniería de software y cómo a través de los años han surgido nuevas metodologías y prácticas con el fin de construir productos que cumplan con las necesidades de los usuarios a un costo y en un tiempo considerable. También se pudo apreciar qué es la calidad de software, su importancia y el papel que juegan las pruebas de software en el aseguramiento de la calidad y, por último, se presentó el concepto, historicidad y estructura del Modelo de Madurez de Capacidad Integrada (CMMI), el cual servirá de referencia para proponer las mejoras del proceso de calidad de software de la empresa Wepsys S.R.L.

De la información recolectada también se puede concluir que el área de calidad es de gran importancia, puesto que tener un proceso de calidad de software funcional abre las puertas a tener un mercado competitivo más amplio donde aparte de adquirir nuevos clientes, se crea fidelidad con los clientes anteriores.

CAPÍTULO II: ASPECTOS METODOLÓGICOS

Introducción

El propósito de este capítulo es explicar los aspectos metodológicos utilizados en la investigación que sirve de base para el desarrollo de este proyecto. En este, se definen los tipos y métodos de investigación aplicados, el diseño de la investigación, la población y muestra y finalmente el tratamiento de los datos recolectados.

2.1 Tipo de investigación

La investigación científica se define como el proceso sistemático de indagación el cual estudia y analiza un hecho, tema o fenómeno, utilizando métodos y criterios determinados con el propósito de proporcionar una explicación o resolver una problemática.

En este proyecto el objetivo principal es proponer mejoras al proceso de calidad de software de la empresa Wepsys S.R.L. De acuerdo con su enfoque esta investigación es social y proyectiva.

De la misma manera esta investigación también se puede clasificar en base a las técnicas de recolección de datos utilizadas, en este caso estas son la investigación documental y la investigación de campo.

2.1.1 Investigación social

El concepto de investigación social se refiere al proceso de generar conocimiento que se encuentre relacionado con las realidades y necesidades sociales, uno de sus focos principales es describir una problemática y sus causas u origen.

Este es el carácter de investigación que corresponde con este proyecto investigativo, ya que la investigación social es el método utilizado para realizar el diagnóstico del estado actual del proceso de calidad de software y proponer mejoras al mismo y con esto poder satisfacer las necesidades sociales.

2.1.2 Investigación proyectiva

Consiste en elaborar una propuesta que permita solucionar problemas o necesidades de tipo práctico para un grupo social, área o institución, a partir de un diagnóstico de una necesidad del momento.

Otras definiciones afirman que este enfoque tiene como objetivo diseñar o crear respuestas dirigidas a determinadas situaciones mediante un proyecto (Córdoba & Monsalve).

Este tipo de investigación consta de las siguientes fases:

- **Fase exploratoria:** en esta fase se detecta la presencia de estudios descriptivos y eventos a modificar.
- **Fase descriptiva:** se describe la situación en cuestión.
- Fase comparativa: se realizan investigaciones del evento a modificar y sus posibles causas.
- **Fase analítica:** se analizan las teorías del evento a cambiar y sus causas.
- **Fase explicativa:** se describe el contexto y las causas.
- **Fase predictiva:** se estudia la factibilidad del cambio y las dificultades y limitaciones, también se ajustan los objetivos.
- **Fase proyectiva:** se diseña el proyecto, se seleccionan las unidades de estudio, se elaboran instrumentos de diagnóstico.
- **Fase interactiva:** se aplican los instrumentos y se recogen los datos.
- **Fase confirmatoria:** se analiza y se construye el diseño, propuesta o plan de acción del cambio.
- **Fase evaluativa:** se realizan las recomendaciones de lugar.

Según la descripción de la investigación proyectiva, este proyecto cuenta con todas las características de este tipo de investigación, ya que dentro de sus objetivos está el diagnóstico de una situación y la propuesta de mejoras.

2.1.3 Investigación documental

Se describe como una técnica de recolección de datos, mediante la cual se selecciona y recopila información de documentos, libros, revistas, periódicos, grabaciones, etc.

Dentro de este tipo de investigación se encuentra la información documental exploratoria, la cual se encarga de probar si algo es correcto o incorrecto, y hallar soluciones luego de hacer una evaluación de las informaciones recolectadas.

En este caso, contamos con la investigación documental debido a que la fuente principal para proponer las mejoras al proceso es la documentación oficial del Modelo de Madurez

de Capacidad Integrado (CMMI) en su versión 1.3, enfocándonos específicamente en el área de proceso de Aseguramiento de la Calidad de Procesos y Productos (PPQA).

2.1.4 Investigación de campo

Este tipo de investigación consiste en la adquisición y medición de datos de un suceso en particular y en el lugar donde sucede.

Para evaluar el proceso de calidad de software fue necesario trasladarnos a la empresa Wepsys S.R.L. con el objetivo de recolectar información veraz, precisa y útil para la investigación.

2.2 Diseño de la investigación

El diseño de investigación es el marco que brinda la dirección y el plan del investigador para obtener las respuestas necesarias a las interrogantes planteadas.

Para realizar este proyecto, se diseñó una encuesta que tiene como objetivo determinar la percepción del personal de Wepsys acerca del proceso de Calidad de Software en específico y de áreas que inciden directamente en el mismo (Ver Anexo 1).

La encuesta está compuesta por un total de 24 preguntas, de estas, existen 10 preguntas que tienen como objetivo conocer la población y el tipo de producto construido por Wepsys, 13 preguntas cerradas para evaluar el nivel de conformidad relativo al proceso de calidad de software actual, en esta sección se utilizó la escala de Likert, la cual permite medir aspectos cualitativos y en adición una pregunta abierta con el objetivo de obtener posibles propuestas de mejora en el proceso.

Ya que el objetivo es evaluar el estado actual del proceso, se utilizó una investigación de campo no experimental, ya que no se pretende modificar las condiciones ni los resultados de la investigación, sino, observar y evaluar los hechos tal cual suceden.

2.3 Métodos de investigación

El método científico ha sido definido de diversas maneras. Algunos autores lo precisan como un procedimiento para tratar un conjunto de problemas. Otros lo han definido como un procedimiento racional e inteligente de dar respuesta a una serie de incógnitas

entendiendo su origen su esencia y su relación con uno o varios efectos. Los tipos de métodos utilizados en nuestro trabajo se investigación son:

2.3.1 Método Inductivo

El método inductivo es una estrategia de razonamiento que se utiliza para poder obtener conclusiones generales partiendo de hechos particulares. Es el método científico más usado.

El método inductivo es aquel procedimiento de investigación que pone en práctica el pensamiento o razonamiento inductivo. Este último se caracteriza por ser ampliativo, o sea, generalizador, ya que parte de premisas cuya verdad apoya la conclusión, pero no la garantiza.

El razonamiento inductivo consiste, así, en una forma de hipótesis que, a partir de una evidencia singular, sugiere la posibilidad de una conclusión universal. Esto suele expresarse en términos de probabilidades, tendencias o posibilidades, ya que no es posible afirmar nada de manera rotunda, ya que existe más información vital que la contenida en las premisas.

Esta forma de razonamiento es muy valiosa, dado que incorpora la creatividad y permite arriesgar conclusiones innovadoras que, si bien no pueden demostrarse, sí pueden someterse a consideraciones, pruebas y mecanismos de validación que, posteriormente, conduzcan a la verdad. Por eso, el método inductivo forma parte del método científico, dado que sirve para expandir el conocimiento del mundo real que tenemos.

El origen moderno del método inductivo se remonta al siglo XVII y a la obra del filósofo inglés sir Francis Bacon (1561-1626), particularmente a su Novum organum scientiarum (“Nuevos instrumentos científicos”) de 1620, donde precisó las reglas del método científico.

Se contrapuso a la tradición aristotélica del momento, en la que sólo se valoraban los razonamientos deductivos. Así, Bacon intentó demostrar la importancia de los razonamientos inductivos, pero aclarando que para llegar a una conclusión es necesario excluir muchas otras posibilidades.

Filósofos posteriores como David Hume (1711-1776), John Herschel (1792-1871) y John Stuart Mill (1806-1873) continuaron la tradición inaugurada por Bacon, y propusieron distintas formas de plantear la inducción con fines rigurosamente científicos.

Dicho esto, este método es ideal para realizar un diagnóstico acerca del estado actual del proceso de calidad de software de Wepsys, debido a que desde distintas informaciones y premisas se va a elaborar una conclusión.

2.3.2 Método Deductivo

Se habla del método deductivo para referirse a una forma específica de pensamiento o razonamiento, que extrae conclusiones lógicas y válidas a partir de un conjunto dado de premisas o proposiciones. Dicho de otra forma, es un modo de pensamiento que va de lo más general (como leyes y principios) a lo más específico (hechos concretos).

Según este modo de pensamiento, las conclusiones de un razonamiento están dadas de antemano en sus propias premisas, por lo que sólo se requiere de un análisis o desglose de éstas para conocer el resultado. Para poder hacerlo, las premisas deben darse por verdaderas, ya que de su validez dependerá que las conclusiones sean o no verdaderas también.

El método deductivo puede emplearse de dos maneras:

Directa. En este caso se parte de una única premisa que no es contrastada con otras a su alrededor.

Indirecta. En este caso se parte de un par de premisas: la primera contiene una afirmación universal y la segunda una particular; de la comparación de ambas se obtiene la conclusión.

De esta manera, la validez de las premisas determinará si las conclusiones también son válidas.

Por otro lado, este método da pie a dos otros métodos más complejos, que son:

Método axiomático-deductivo. Se extrae un conjunto de teoremas (proposiciones) a partir de un conjunto de axiomas (premisas) dados de antemano, empleando series de razonamientos lógicos.

Método hipotético-deductivo. A partir de la observación de un fenómeno, se aventura una hipótesis interpretativa que luego se somete a comparación por razonamientos lógicos de tipo deductivo. Este es el método que emplea el conocimiento científico.

El método deductivo es el ideal para realizar las recomendaciones de mejora al proceso de calidad de software a partir del estado obtenido.

2.3.3 Método Cuantitativo

El método cuantitativo es aquel que se basa en medición y estadística para investigar, analizar y comprobar información y datos.

Los datos cuantitativos se definen como aquellos que se muestran de manera numérica como porcentajes y estadísticas, esto implica que la investigación cuantitativa utiliza preguntas específicas y obtiene muestras numéricas por medio de las respuestas a esas preguntas. Una técnica cuantitativa muy utilizada es la encuesta, la cual es un conjunto de preguntas que tiene la finalidad de obtener datos de una muestra perteneciente a una población de interés.

La encuesta fue el medio utilizado para obtener la información que permita determinar el nivel de conformidad de los empleados con el actual proceso de calidad de software de Wepsys S.R.L.

2.4 Población y Muestra

2.4.1 Población

Según Aura Bavaresco (2015), "Una población es la totalidad de los elementos que forman un conjunto". Por otra parte, Ludewig, (2016), indica que "Una población es cualquier colección finita o infinita de elementos o sujeta".

La población de análisis que participó en la investigación corresponde a los empleados del área de desarrollo, proyecto y control de la calidad de WEPSYS SRL.

El total de los empleados de las áreas de desarrollo y proyecto son 28 personas.

Se aplicó la encuesta a los empleados en el mes de octubre 2020, con la finalidad de conocer cómo se lleva el proceso de control de calidad dentro de las instalaciones de wepsys SRL.

2.4.2 Muestra

Para determinar la muestra se decidió implementar el método aleatorio simple para poblaciones finitas. En este método se deben tomar en cuenta las siguientes variables:

- **Tamaño de la población:** representada por la letra N. De acuerdo con el punto 2.4.1 la población es de 28 personas.
- **Margen de error:** identificado por la letra E, su valor es de 5%.

- **Margen de seguridad:** identificado por la letra Z, su valor es de 95% y su coeficiente equivale a 1.96 según la tabla de distribución estándar.
- **Desviación estándar:** representado por la letra S, su valor es de un 50%.

La fórmula establecida para determinar la muestra es la siguiente:

$$n = \frac{S^2}{\frac{E^2}{Z^2} + \frac{S^2}{N}}$$

Conociendo todas las variables y la fórmula, se procedió a sustituir los datos:

Datos	
S	0.5
E	0.05
Z	1.96
N	28

$$\boldsymbol{n} = \frac{0.25}{\frac{0.0025}{3.8416} + \frac{0.25}{28}} ; \boldsymbol{n} = \frac{0.25}{0.0089} ; \boldsymbol{n} = 28$$

En última instancia, la muestra representa una porción idéntica a la población, así que se tomó el 100% de la población para realizar la encuesta.

2.5 Tratamiento de la información

2.5.1 Tabulación

Luego de construir la matriz de los datos y aplicar las encuestas, se procedió al análisis y posteriormente a la tabulación de los datos. Dichos datos fueron plasmados en valores enteros positivos.

2.5.2 Gráficas

Una vez obtenidas las tabulaciones de los datos recopilados en la investigación, se procedió a graficar dichos datos, utilizando gráficas de pastel.

Resumen del capítulo II

En este capítulo se definieron los aspectos metodológicos utilizados para llevar a cabo la investigación en cuestión. Dentro de estos, se destacan el diseño de la investigación, la población y la muestra, así como también, los tipos de investigación utilizados con el fin de determinar el estado actual del proceso de calidad de software de la empresa Wepsys y de proponer mejoras al mismo.

Entre los tipos de investigación utilizados están: la investigación social, proyectiva, documental y de campo. También se definieron los métodos utilizados, estos son: método inductivo, deductivo y cuantitativo, este último juega un papel importante en la investigación, debido a que define la técnica de recolección de datos utilizada, es decir, una encuesta de 24 preguntas aplicada a los empleados de Wepsys, la cual tiene por objetivo determinar el nivel de satisfacción del proceso de calidad de desarrollo de software de dicha empresa.

CAPÍTULO III: ANÁLISIS DETALLADO SOBRE EL ESTADO ACTUAL DEL PROCESO DE CALIDAD DE SOFTWARE

Introducción

Este capítulo tiene como objetivo dar a conocer el estado actual del proceso de calidad de software de la empresa Wepsys. En este, se muestra la descripción general de la empresa, como su misión, visión, valores, organigrama y actividades clave, también se define el análisis FODA, la descripción y diseño del proceso actual de calidad de software y la presentación y análisis de los resultados de la encuesta aplicada. Todo esto en conjunto es de vital importancia para determinar el estado actual del proceso antes mencionado.

3.1 Descripción de Wepsys S.R.L

3.1.1 Historia

Wepsys S.R.L. es una factoría de software fundada en el 2007, por el Ing. Pablo Lorenzo, el Ing. William Werner y el Ing. Stalin Rivas. Desde el 2007 se dedica al desarrollo de software de manera ininterrumpida, colaborando con varios clientes tanto en el sector privado como público. Ha sido reconocido por Microsoft como Gold Partner en Application Builder.

A principios del 2011 se realizó una alianza estratégica con Novosit, en la cual Wepsys se encargaría de las integraciones de sistemas como ProDoctivity, Onbase y Capture web. Esto hizo que la empresa creciera de manera constante.

A partir del 2015 fue adquiriendo clientes de renombre, como ALNAP, BanReservas, BHD LEÓN, ARS Humano, entre otros. A pesar de que en ese entonces solo eran 6 miembros en la empresa, los proyectos se completaron casi siempre a tiempo, esta característica fue de agrado para sus clientes. También en ese tiempo se agregó el servicio de outsourcing, el cual fue bien recibido, ya que los recursos tenían una calidad sobresaliente.

Con el crecimiento obtenido en los últimos 5 años, Wepsys S.R.L. ha implementado diferentes procesos del manejo del ciclo de vida ALM (Application Lifecycle Management), del cual las pruebas de software ocupan un importante lugar, porque es lo que asegura la calidad de sus productos.

3.1.2 Misión

La empresa Wepsys SRL tiene como misión, brindar soluciones en materia de desarrollo de software, con miras a contribuir en el logro de los objetivos estratégicos de nuestros clientes.

3.1.3 Visión

La empresa Wepsys SRL tiene como visión, ser la empresa de referencia local e internacional en cuanto a la excelencia técnica en materia de desarrollo de software, anticipándonos a los cambios tecnológicos y las necesidades de nuestros clientes, mediante la provisión de oportunidades de crecimiento a nuestros colaboradores dentro de un excelente clima organizacional.

3.1.4 Valores

- Trabajo en Equipo
- Honestidad
- Ética
- Calidad
- Compromiso
- Mejora continua
- Innovación
- Colaboración con el Cliente

3.1.5 Estructura organizacional

La empresa de desarrollo de software Wepsys S.R.L. cuenta con la siguiente estructura organizacional (Ver figura 3.1).

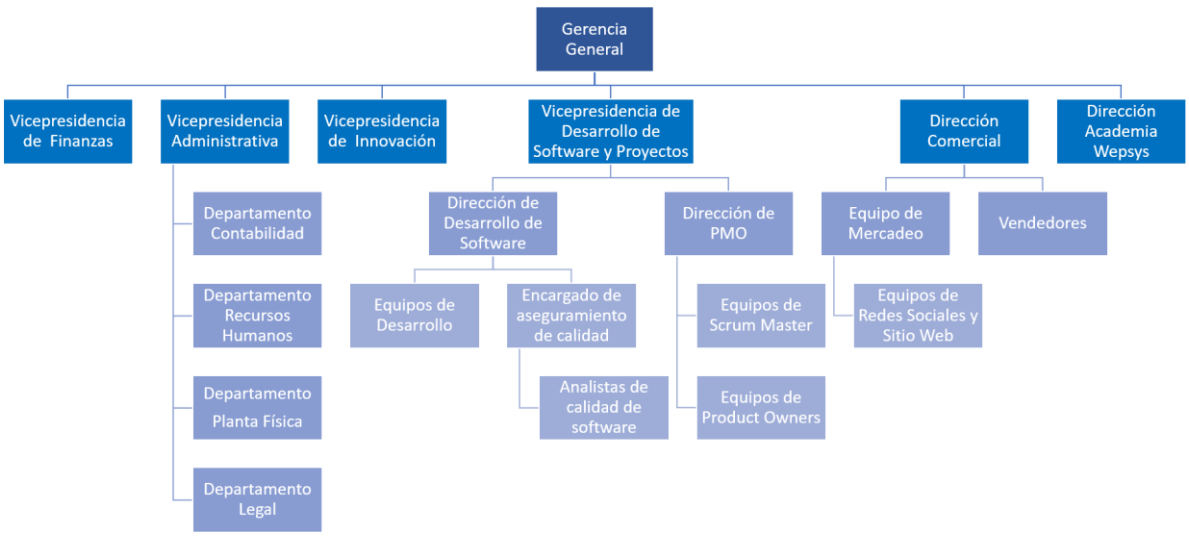


Figura 3.1 - Organigrama de la empresa Wepsys S.R.L.

3.2 Descripción de las actividades principales de Wepsys S.R.L.

La actividad principal de la empresa Wepsys es el desarrollo de aplicaciones personalizadas. Las aplicaciones pueden ser web o móviles (iOS/Android), por lo general desarrolladas en .Net o JavaScript y basadas en Amazon Web Services (AWS).

Wepsys también construye microservicios y web services y brinda servicios de outsourcing. Se caracteriza por ser una empresa orientada a las metodologías ágiles como Scrum y eXtreme Programming.

3.2.1 Análisis FODA

El análisis FODA o DAFO (fortalezas, oportunidades, debilidades y amenazas), es una herramienta de planificación estratégica que sirve para identificar las líneas de acción y planes estratégicos que son necesarios para alcanzar los objetivos de una empresa.

Uno de los objetivos de Wepsys es brindar un servicio de calidad a sus clientes, para conocer los esfuerzos necesarios que permitan cumplir este objetivo, se realizó un análisis FODA, el cual muestra las fortalezas, oportunidades, debilidades y amenazas de la empresa (Ver figura 3.2)

Figura 3.2 - Análisis FODA de Wepsys S.R.L.

3.3 Proceso actual del ciclo de vida del desarrollo del software de Wepsys S.R.L.

3.3.1 Descripción del proceso

En Wepsys SRL el ciclo de vida de desarrollo comienza en la Project Manager Office (PMO), donde se encargan de recibir los requisitos del cliente para luego transformarlos en historias de usuarios basados en la metodología scrum. Esta metodología junto a eXtreme Programing dominan el ciclo de vida del desarrollo completo en Wepsys.

Luego que los requisitos son transformados en historias, estas pasan a priorizarse en un product backlog, al final la realización de las historias determinará el producto que se le entregará al cliente. Esta tarea es realizada por los PM (Project Manager) que a su vez son product owners.

Se utiliza Scrum + eXtreme programing como metodología de trabajo. Los miembros del equipo se dividen en seis, en los cuales hay tres desarrolladores, un QA, un scrum master y un product owner.

Las iteraciones de cada sprint son de dos semanas. Comenzando con la ceremonia de sprint planning, la cual tiene una duración de cuatro horas con dos partes que son time boxed. En esta ceremonia, se les da un peso en history points, cada history point (que es la unidad de esfuerzo) es diferente entre los equipos, ya que estos representan el tiempo singular en que se realizó la tarea más fácil dentro del proyecto.

Desde que se termina la primera parte del planning, el equipo se reúne otras dos horas para crear las tareas técnicas necesarias para realizar la historia de usuario. Allí se encargan de poner todos los escenarios de pruebas de unitarias que se probarán, los cambios del UI o cualquier cambio técnico que no esté escrito en la historia de usuario (Ver figura 3.3).

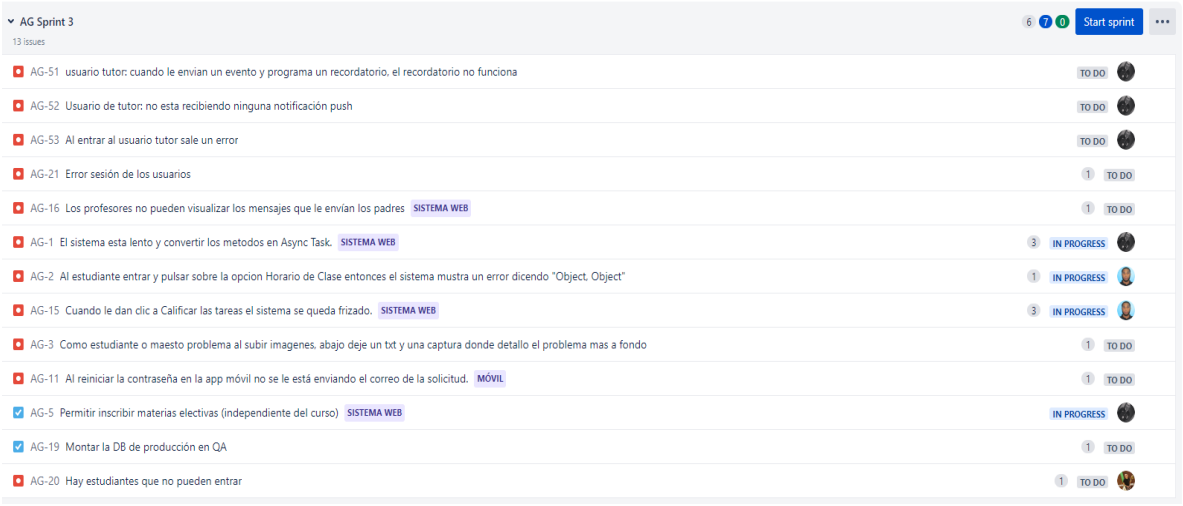


Figura 3.3 - Sprint backlog

A continuación, cada miembro del equipo toma una historia de usuario y procede a trabajarla. El nivel de avance se comenta todos los días a las 9:00 am en la ceremonia de daily meeting de Scrum, aquí cada miembro del equipo dice que hizo el día anterior, que hará en el día actual y si tiene algún impedimento. Esta reunión no suele durar más de quince minutos. Si algún miembro del equipo tiene algún impedimento el scrum master se encarga de anotarlo y resolverlo en la mayor brevedad posible. El equipo utiliza un tablero Kanban, el cual contiene la definición de Done del equipo. Esta definición está compuesta por lo general por las siguientes columnas:

1. **To-Do:** Aquí es donde las tareas comienzan.
2. **In Progress:** Se ponen todas las tareas que se están realizando en el momento.
3. **QA:** Se realiza la verificación de que lo realizado es lo que se pidió y que no afectó otras partes.
4. **Demo:** Se colocan las tareas que serán presentadas de forma individual al Project Owner, para aceptación, cambios o sugerencias.
5. **Pull Request:** Luego de hacer la demo del feature, el desarrollador pasa a crear un pull request donde se revisará el código realizado, sugiriendo refactor del código.
6. **Done:** Aquí se da por terminada la historia de usuario.

Esta es la definición base de la cual cada equipo parte (Ver figura 3.4).

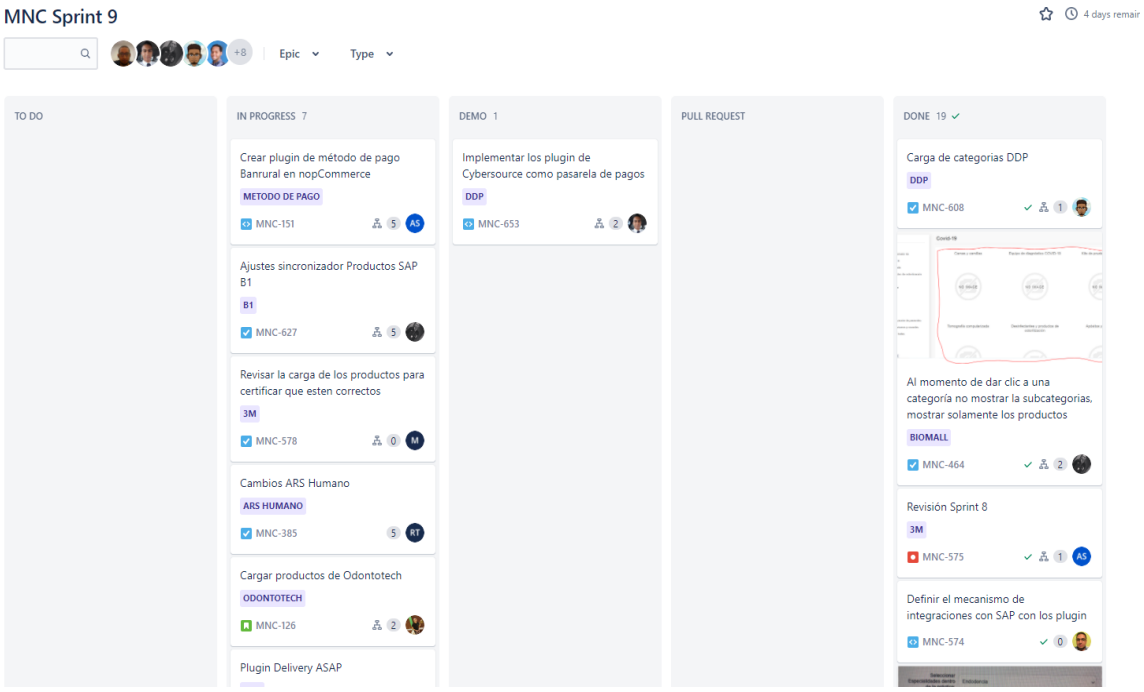


Figura 3.4 - Kanban de actividades

A mitad del sprint se realiza un grooming que es donde el equipo se reúne con el product owner para granularizar los detalles del product backlog. Esta reunión suele durar dos horas, la cual sirve para detallar y priorizar los ítems del product backlog.

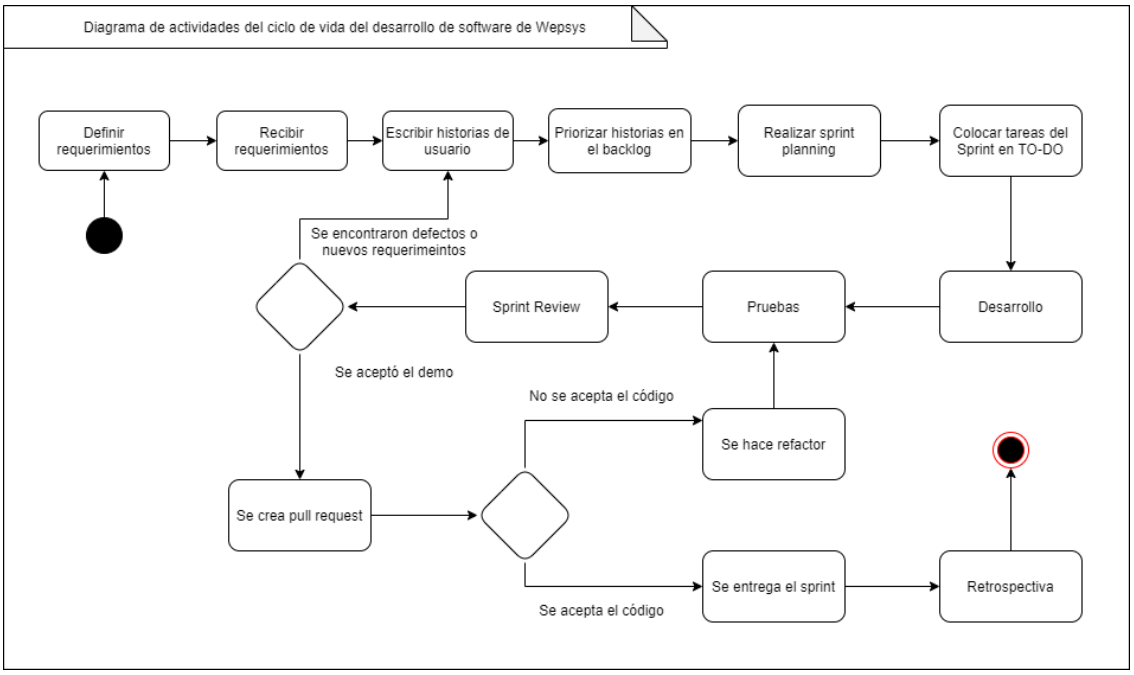
Al final del sprint se realiza la ceremonia del sprint review, donde participan todos los miembros del equipo. Aquí se muestra un demo de las funcionalidades que se realizaron durante el sprint. Dichas funciones quedan aceptadas si el product owner considera que cumplen con los criterios de aceptación que se definieron al inicio del sprint. Esta ceremonia tiene una duración de dos horas. Cuando existen sugerencias por parte del product owner o algún miembro del equipo, estas se escriben de inmediato en el producto backlog y pasan directo al siguiente sprint. Lo mismo sucede con las historias que no se llegaron a completar durante el sprint.

Por último, se realiza la ceremonia de retrospectiva, donde los integrantes del equipo se reúnen para discutir sobre lo que se hizo bien, lo que se hizo mal y lo que se pudo mejorar durante el sprint. De aquí surgen nuevas ideas de convivencia para arreglar lo malo y mantener lo que se está haciendo bien en el próximo sprint.

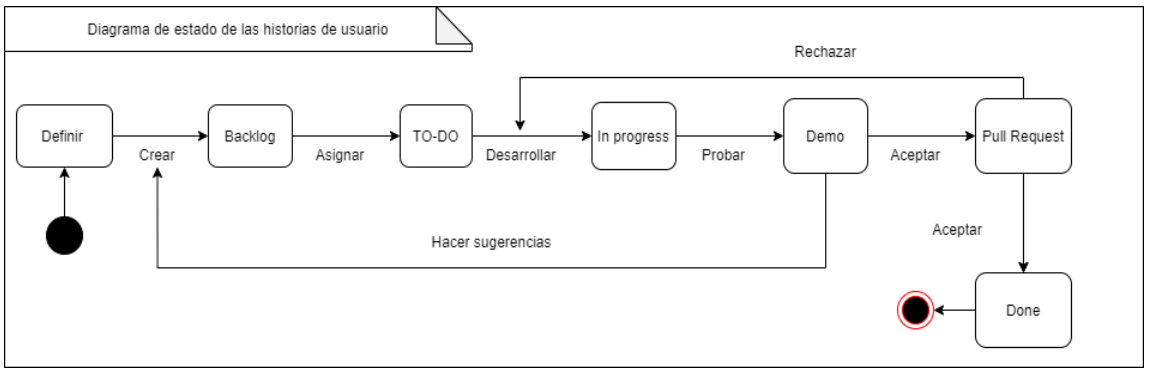
El entregable derivado del esfuerzo que el equipo realizó durante el sprint, es entregado al cliente como incremento de su producto. Esta iteración se repite las cantidades de veces necesarias para terminar el producto.

3.3.2 Diseño del proceso

3.3.2.1 Diagrama de actividades del ciclo de vida de desarrollo de software de Wepsys



3.3.2.2 Diagrama de estado de las historias de usuario



La figura 3.5 muestra la representación gráfica del ciclo de vida de desarrollo en Wepsys, basado en la metodología Scrum.

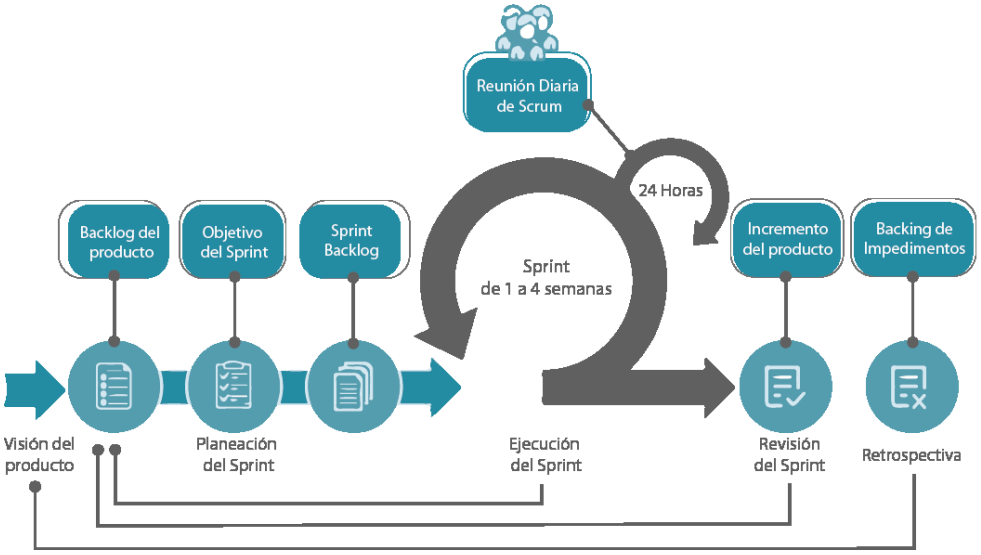


Figura 3.5 Proceso de Desarrollo de software

3.4 Presentación y análisis de los resultados

Introducción

Aquí se presentan los resultados de los datos recopilados de manera gráfica y textual mediante el instrumento de recopilación de datos seleccionado para esta investigación. El instrumento seleccionado fue la encuesta, la cual fue aplicada a los empleados Wepsys S.R.L. Esta fue realizada a 28 empleados que están directamente involucrados en el proceso de desarrollo en la empresa.

Estos datos fueron analizados con el propósito de conocer la factibilidad y aceptación de esta investigación.

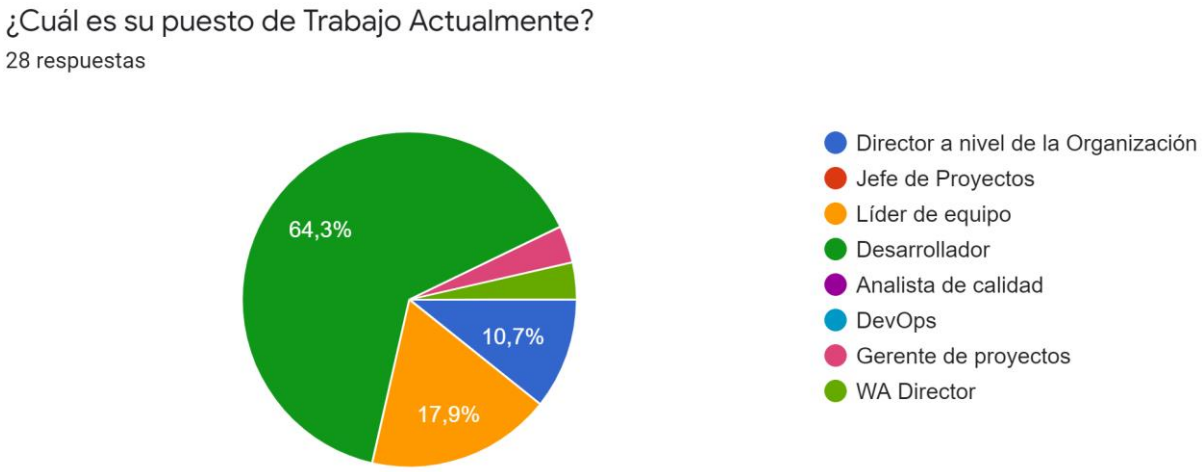


Figura 3.6 - Agrupación por puesto de trabajo de población encuestada

Figura 3.7 - Agrupación por actividad anteriormente realizada.

Aquellos que respondieron sí, especificaron las siguientes actividades:

- Soporte a sistemas
- Analista de requerimientos, QA, desarrollador, product owner
- PO
- Ayudar con los planes de carrera del personal
- Instructor en varias universidades
- Soporte de aplicación ProDoctivity
- Líder de Equipo

Figura 3.8 - Agrupación por cantidad de años en la organización

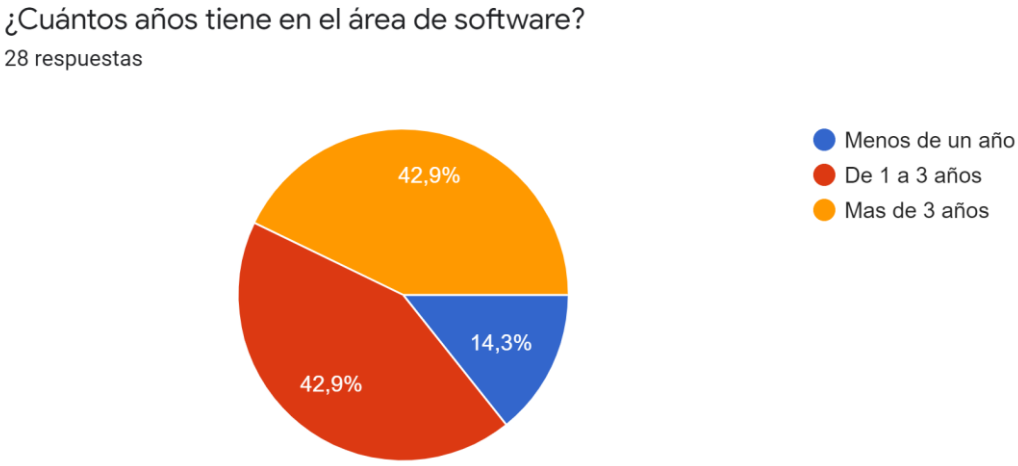


Figura 3.9 - Agrupación por cantidad de años en el área de software

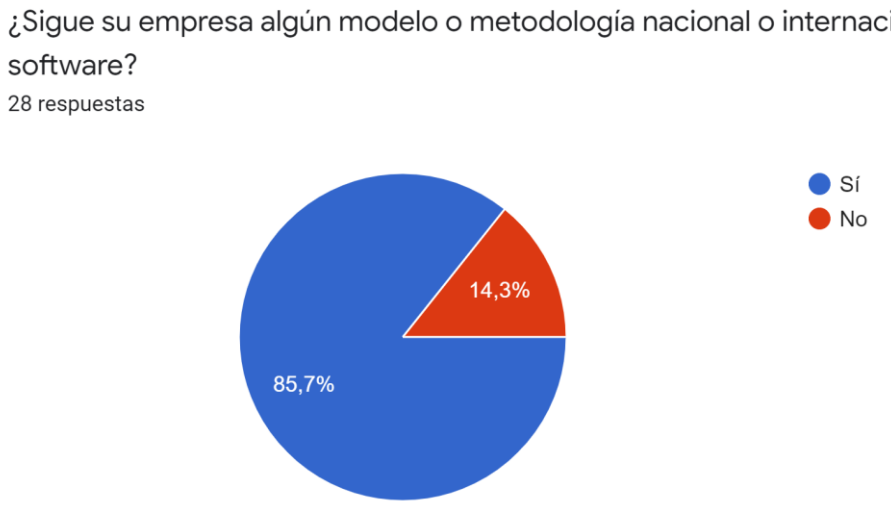


Figura 3.10 - Agrupación por si la empresa usa algún tipo de metodología

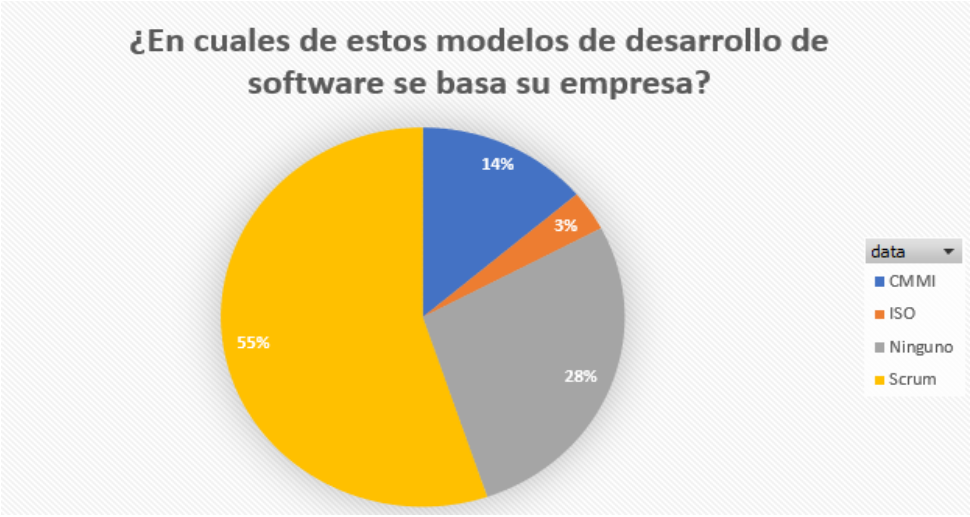


Figura 3.11 - Agrupación de modelos de desarrollo utilizados

Se puede observar que un 55% de los empleados de las áreas de desarrollo y gestión de proyectos de Wepsys, asegura que la empresa utiliza Scrum como modelo de desarrollo.

Figura 3.12 - Agrupación de conocimiento de modelos de software

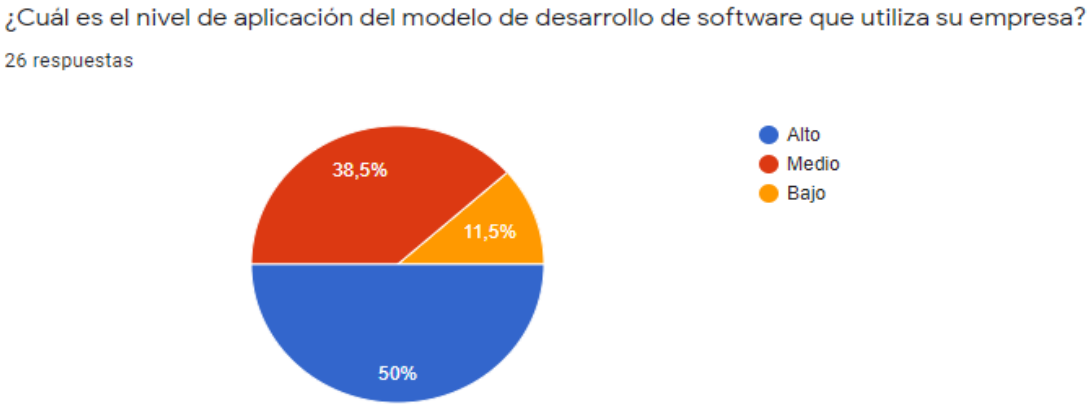


Figura 3.13 - Agrupación de nivel de modelo de desarrollo de software

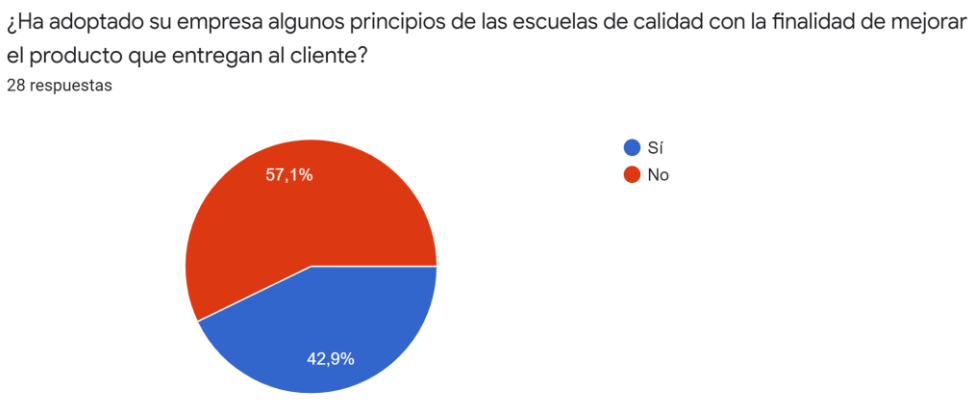


Figura 3.14 - Agrupación de adopción de principios de las escuelas de calidad de software con finalidad de mejorar el producto

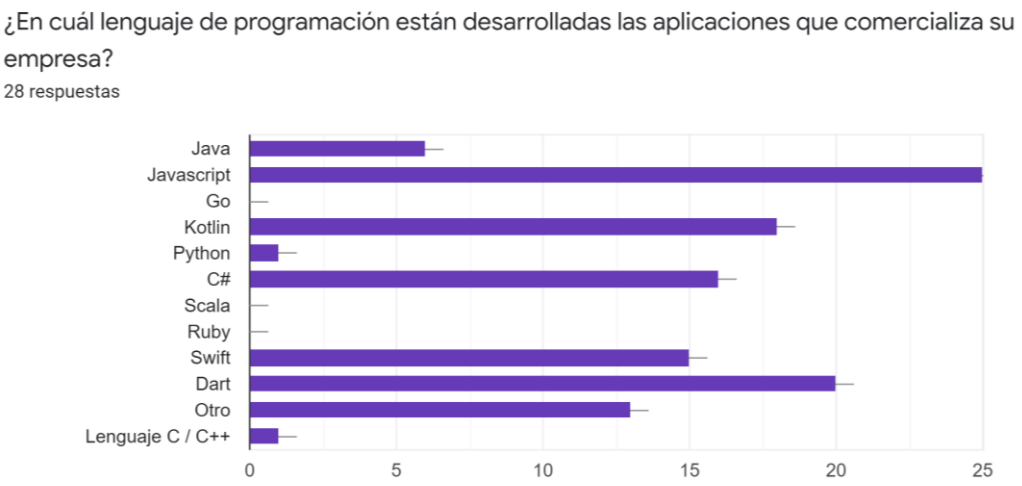


Figura 3.15 - Agrupación de lenguajes de programación utilizados

En el caso de base de datos, el 100% respondió SQL Server.

Figura 3.16 - Porcentajes del nivel de satisfacción: ¿Los requerimientos de sistema asignados al software son usados para establecer una Línea Base para el uso de la Ingeniería y Gestión de Software?

La figura 3.16 muestra que de los 28 encuestados, el 64% de respuestas fueron positivas (43% de acuerdo y 21% totalmente de acuerdo) con relación a que los requerimientos del sistema son utilizados como base de la ingeniería y gestión de software, mientras que el 25% tiene una posición neutral y solo un 11% está en desacuerdo o en total desacuerdo.

Figura 3.17 - Porcentajes del nivel de satisfacción: ¿Cuándo los requerimientos del sistema cambian, se hacen los ajustes necesarios a los planes, productos de trabajo de software y otras actividades relacionadas?

De acuerdo con la figura 3.17, un 61% (36% totalmente de acuerdo y 25% de acuerdo) de los 28 empleados encuestados tuvo una respuesta positiva acerca de que cuando los requerimientos de software cambian, los demás artefactos de software son actualizados según correspondan. Un 25% de empleados mantuvo una posición neutral, mientras que un 14% está en desacuerdo con este planteamiento.

Según esta pregunta y la anterior, más de la mitad de los empleados se encuentran satisfechos con la gestión de requerimientos.

Figura 3.18 - Porcentajes del nivel de satisfacción: ¿El proyecto sigue una política organizacional escrita para la gestión de los requerimientos de sistema asignados al software?

De acuerdo con la figura 3.18, solo el 39% (25% totalmente de acuerdo y 14% de acuerdo) afirma que se sigue una política organizacional para la gestión de requerimientos de sistema. El 61% restante, no está de acuerdo o seguro de si se sigue una política organizacional para la gestión de requerimientos.

Esto significa que, aunque según las figuras 3.16 y 3.17, los empleados tienen una percepción positiva con relación a la gestión de requerimientos, de acuerdo a los resultados de esta pregunta, este procedimiento se realiza de manera informal, pues no existe o no se sigue una política organizacional para la gestión de los requisitos.

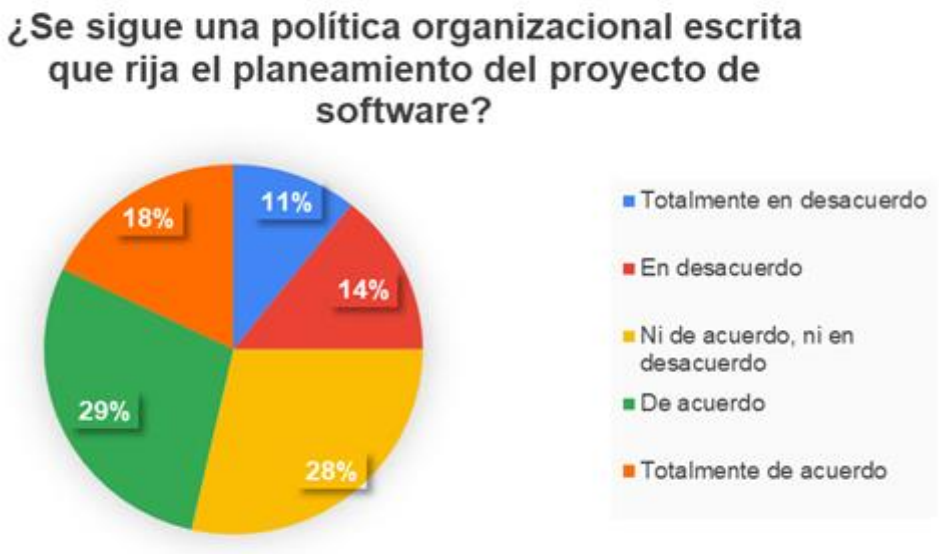


Figura 3.19 - Porcentajes del nivel de satisfacción: ¿Se sigue una política organizacional escrita que rija el planeamiento del proyecto de software?

Según la figura 3.19, el 47% está de acuerdo en que se sigue una política organizacional como base para el planeamiento de los proyectos, mientras que un 28% no está ni acuerdo ni en desacuerdo, un 25% se encuentra en desacuerdo.

Figura 3.20 - Porcentajes del nivel de satisfacción: ¿Se planifican las actividades de Aseguramiento de la calidad de software?

La figura 3.20 muestra que un 36% de los empleados del área de desarrollo y proyectos está totalmente de acuerdo y un 25% está de acuerdo con que las actividades de aseguramiento de calidad son planificadas, esto da una respuesta favorable de un 61%, un 14% no se encuentra de acuerdo, ni en desacuerdo, mientras que un 14% dice estar en desacuerdo y un 11% totalmente en desacuerdo.

Figura 3.21 - Porcentajes del nivel de satisfacción: ¿El aseguramiento de la calidad suministra una verificación objetiva de que las actividades y productos de software se corresponden con los procedimientos, estándares aplicables y con los requerimientos?

El 36% de los encuestados está totalmente de acuerdo con que se suministra una verificación objetiva de que las actividades y productos se corresponden con los procedimientos y con los requerimientos, de la misma manera un 28% está de acuerdo, un 18% se encuentra neutral al respecto, un 11% y un 7% no se encuentran de acuerdo o totalmente de acuerdo con el planteamiento (Ver figura 3.21).

Figura 3.22 - Porcentajes del nivel de satisfacción: ¿El proyecto sigue una política organizacional escrita tanto para el seguimiento como para el control de las actividades de desarrollo del software?

De acuerdo con la figura 3.22, un 54% (29% de acuerdo y 25% totalmente de acuerdo) de los empleados encuestados, de los cuales la mayoría son desarrolladores de software, considera que se sigue una política organizacional escrita para las actividades de desarrollo. Un 28% adoptó una posición neutral, mientras que un 11% se encuentra en desacuerdo y un 7% está totalmente en desacuerdo.

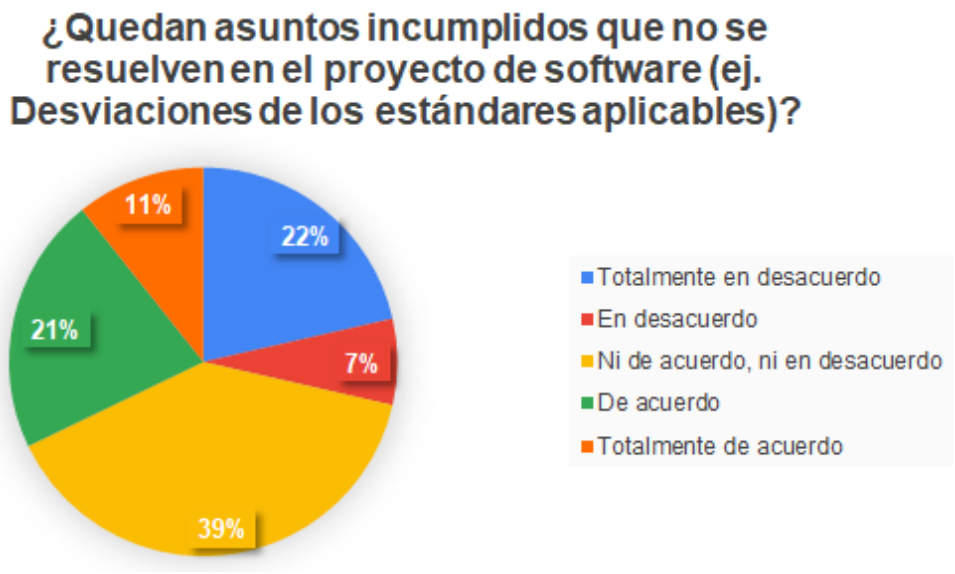


Figura 3.23 - Porcentajes del nivel de satisfacción: ¿Quedan asuntos incumplidos que no se resuelven en el proyecto de software (ej. Desviaciones de los estándares aplicables)?

Según la figura 3.23, solo el 29% de los encuestados no está de acuerdo con que se quedan asuntos incumplidos que no se resuelven en los proyectos, mientras que un 32% está de acuerdo en que sí se quedan asuntos incumplidos sin resolver y un 39% sostiene una posición neutral al respecto.

Figura 3.24 - Porcentajes del nivel de satisfacción: ¿El proyecto sigue una política organizacional escrita y aprobada por la máxima autoridad para la implementación del aseguramiento de la calidad?

De acuerdo con la figura 3.24, solo la mitad de los empleados encuestados (25% de acuerdo y 25% totalmente de acuerdo), considera que se sigue una política organizacional escrita y aprobada por la máxima autoridad para la implementación del aseguramiento de calidad, mientras que el 50% restante no está de acuerdo o no está seguro.

Figura 3.25 - Porcentajes del nivel de satisfacción: ¿Las actividades de aseguramiento de la calidad cuentan con los recursos adecuados (¿ej. fondos y personas entrenadas, personas que dirijan y se responsabilice con la actividad?

Según la figura 3.25, solo un 46% está de acuerdo o totalmente de acuerdo con que las actividades de calidad cuentan con los recursos adecuados, mientras que un 29% no está ni de acuerdo, ni en desacuerdo y un 25% se encuentra en desacuerdo o totalmente en desacuerdo.

Si analizamos las figuras 3.6 y 3.7, se puede destacar que el personal que se encarga de las actividades específicas de calidad también comparte su tiempo realizando otras actividades, esto quiere decir que si bien hay recursos que se encargan de llevar el proceso de calidad, no existen recursos 100% dedicados a estas tareas.

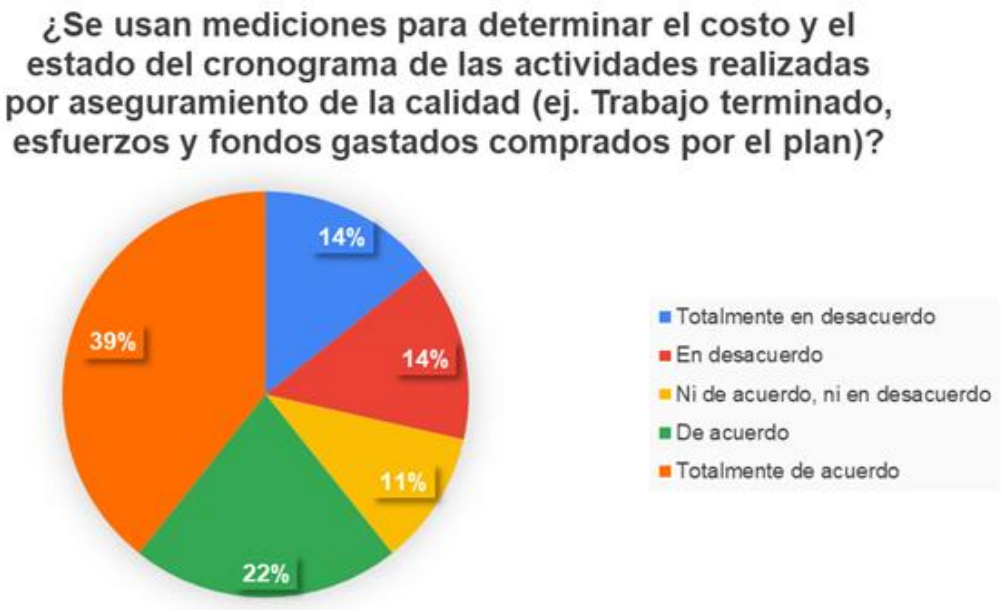


Figura 3.26 - Porcentajes del nivel de satisfacción: ¿Se usan mediciones para determinar el costo y el estado del cronograma de las actividades realizadas por aseguramiento de la calidad (ej. Trabajo terminado, esfuerzos y fondos gastados comprados por el plan)?

En la figura 3.26 se muestra que el 39% está totalmente de acuerdo con que se utilizan mediciones para determinar costos y estado del cronograma de las actividades realizadas, el 22% está de acuerdo, para un 61% de respuestas favorables, mientras que un 28% se reparte a mitad entre en desacuerdo y totalmente en desacuerdo y un 11% no se encuentra ni de acuerdo, ni en desacuerdo.

Figura 3.27 - Porcentajes del nivel de satisfacción: ¿Las actividades de aseguramiento de la calidad son revisadas periódicamente?

De acuerdo con la figura 3.27, solo el 47% de los colaboradores encuestados está de acuerdo con que las actividades de aseguramiento de calidad son revisadas periódicamente, mientras que el 53% restante, no está de acuerdo o no está seguro con que estas actividades se revisan periódicamente.

Figura 3.28 - Porcentajes del nivel de satisfacción: ¿Cree usted que las prácticas recomendadas por CMMI ayudarían a mejorar el proceso de Calidad de Software en su empresa?

De acuerdo con la figura 3.28, un 61% de los empleados de las áreas de desarrollo y proyectos entiende que las prácticas recomendadas por el Modelo de Mejora de Capacidad Integrado (CMMI) ayudarían a mejorar el proceso de calidad de software, de este 61%, el 43% está totalmente de acuerdo y un 18% está de acuerdo, mientras que el 25% dice no estar ni acuerdo, ni en desacuerdo y solo un 14% se encuentra entre estar en desacuerdo y totalmente en desacuerdo.

Finalmente, se realizó la siguiente pregunta abierta: **¿Tiene algún comentario acerca de cómo puede ser mejorado el proceso de calidad de Software de la empresa?**

Respuestas recibidas:

- Más cursos y capacitaciones.
- El proceso puede mejorar si se dedican recursos o personas para que se encarguen específicamente de las revisiones y el aseguramiento de la calidad. Tales como un QA y que este cree defectos, por ej. en Jira, para que posteriormente los desarrolladores puedan corregirlos. Con esto se reduce el riesgo de que en los demos se encuentren defectos o peor aún, al momento de entregarle al cliente y que este se decepcione por la funcionalidad esperada.
- Crecimiento del equipo de calidad de software, capacitación y certificación en control de calidad de software en el equipo, definición de reglas y proceso de control de calidad a cumplir en todos los proyectos construidos, el cual debe ser verificado, validado y firmado.
- Formalizar de mejor manera los tiempos de entrega con el cliente.
- Facilitar cursos y certificaciones a los desarrolladores.
- Dar entrenamientos sobre el proceso de calidad de software.
- Que el personal de QA cree y documente los casos de usos que prueban en las aplicaciones, para que, si hay rotación de personal, los nuevos se puedan poner al día rápidamente, y no tener que explicarle toda la aplicación para luego tener la capacidad de probar.

Los resultados obtenidos mediante la encuesta nos llevan a las siguientes conclusiones acerca del estado del proceso de calidad de software:

A. Partiendo de los resultados mostrados en las cuatro preguntas acerca de si se siguen políticas para la gestión de los requerimientos, la planeación de los proyectos, el desarrollo de software y el aseguramiento de la calidad (Ver figuras 3.18, 3.19, 3.22 y 3.24), en promedio, aproximadamente el 31% de los empleados encuestados se mantuvieron neutrales acerca de si se siguen políticas para los cuatro procesos mencionados anteriormente, en el caso de estas preguntas, no estar ni de acuerdo, ni en desacuerdo resulta en una preocupación, debido a que se puede considerar que un tercio de los empleados encuestados sigue de manera irregular estas políticas o no conoce de su existencia.

B. Según las informaciones recolectadas acerca de las posiciones y recursos en general (Ver figuras 3.6, 3.7 y 3.25), el personal que se encarga de realizar las actividades de calidad de software no está dedicado 100% a esta sola área, por tanto, esto trae consecuencias como reducción del tiempo de pruebas, más posibilidad de que se escapen escenarios de prueba, falta de documentación como casos de pruebas.

C. Se puede observar que solo el 47% de los encuestados asegura que las actividades de calidad son revisadas periódicamente.

D. En las propuestas de mejora, la mayoría de las respuestas coincidieron en la falta de personal dedicado a llevar el aseguramiento de la calidad, las capacitaciones al personal y la creación y cumplimiento de políticas y procedimientos de calidad de software.

E. Un 61% del personal encuestado está de acuerdo en que las prácticas recomendadas por CMMI serían de ayuda para mejorar el proceso de calidad de software, solo el 14% estuvo en desacuerdo.

F. A nivel general la percepción del personal de desarrollo y proyectos acerca del proceso de calidad de software resultó en un nivel de satisfacción intermedio, partiendo de las respuestas le podemos asignar un 3/5.

Resumen del capítulo III

En este capítulo se realizó una descripción de la empresa Wepsys en la cual se dio a conocer su historia, misión, visión, valores, estructura organizacional y su actividad principal que es el desarrollo de aplicaciones para plataformas web y móviles. También se realizó un análisis FODA, donde se mostraron las fortalezas, oportunidades, debilidades y amenazas de la empresa, cabe destacar que una de las debilidades de este análisis se trata de que el proceso de calidad de software no se encuentra bien definido.

Se realizó un levantamiento del proceso actual del ciclo de vida del desarrollo de software que se lleva en la empresa, se describieron las actividades que componen este proceso, los roles del personal que las realiza y los artefactos que se utilizan. Este proceso fue representado gráficamente en tres diagramas, los cuales son: un diagrama UML de actividades, un diagrama UML de estados y un diagrama de la aplicación de la metodología Scrum en el proceso de desarrollo.

Con el objetivo de evidenciar la problemática y conocer la percepción de los colaboradores de las áreas involucradas en el proceso de desarrollo de software, se realizó una encuesta de 24 preguntas dividida en dos secciones, la primera se encarga de recolectar información general acerca del área de desarrollo y de sus procesos y la segunda sección tiene como propósito dar a conocer el nivel de percepción de los empleados acerca de las distintas políticas y procedimientos relacionados al proceso de calidad de software.

En la encuesta se determinó que solo el 39% de los encuestados está de acuerdo con que se sigue una política organizacional para la gestión de los requerimientos, la cual es un área que influye directamente en la calidad del producto, solo el 50% asegura que se sigue una política para el aseguramiento de la calidad, este resultado deja abierta la posibilidad de que la mitad de los colaboradores del área de desarrollo y proyectos no siga la política o no conozca de su existencia y el 61% está de acuerdo con que las prácticas recomendadas en el área clave de proceso PPQA de CMMI ayudaría a mejorar el proceso de calidad de software actual.

CAPÍTULO IV: PROPUESTA DE MEJORAS AL PROCESO DE CALIDAD DE SOFTWARE

Introducción

En este capítulo, se describe el área clave de proceso de Aseguramiento de la calidad del proceso y del producto (PPQA), la cual forma parte del Modelo de Madurez de Capacidad Integrado (CMMI) desarrollado por el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon, esta área nos servirá de base para la elaboración de la propuesta de mejora al proceso de calidad de software de la empresa Wepsys. También se presenta la descripción del proceso mejorado.

4.1 Antecedentes

En el capítulo anterior fue mostrado el proceso actual del ciclo de vida de desarrollo de software que la empresa utiliza y se realizó una encuesta de conformidad donde salieron a relucir las siguientes debilidades con respecto al proceso de calidad de software:

- Solo la mitad de los colaboradores entiende que se sigue una política para el aseguramiento de la calidad, sin embargo, en nuestra investigación de campo, pudimos notar que no existe ninguna política escrita para este proceso.
- Las actividades de aseguramiento de la calidad no son revisadas periódicamente.
- Los colaboradores que se encargan de realizar las actividades de calidad de software no están dedicados completamente a esta sola actividad, por tanto, esto trae consecuencias como reducción del tiempo de pruebas, falta de documentación como casos de pruebas y levantamiento de defectos.
- El personal requiere de capacitaciones en temas de calidad de software.

En general, el 61% de los colaboradores pertenecientes al área de desarrollo de software y proyectos, estuvo de acuerdo con que las prácticas recomendadas por CMMI con relación al área clave de proceso de Aseguramiento de la calidad de procesos y de productos, ayudarían a mejorar el proceso de calidad de software de la empresa, por tanto, hemos desarrollado la siguiente propuesta en base al área clave de proceso PPQA.

4.2 Propuesta de mejoras al proceso de calidad de software actual de la empresa Wepsys S.R.L

1. Objetivos

El propósito del área de proceso de aseguramiento de la calidad de procesos y productos es proveer información objetiva sobre los procesos y los productos de trabajo. Es decir, este procedimiento pretende asegurar que todos los procesos y productos definidos estén siendo respetados en la organización, esto con el objetivo de detectar deficiencias en la forma de trabajar ya establecida.

Las metas de esta área de proceso serán evaluar objetivamente la ejecución de los procesos, los elementos de trabajo y servicios en base al cumplimiento de los estándares y procedimientos definidos. Además de identificar y documentar no conformidades, como el incumplimiento de estándares o proceso de la organización

2. Alcance

Este procedimiento define actividades para asegurar la calidad del proceso de desarrollo de software mediante un ambiente que cumpla con los estándares de calidad de software. Este ambiente contiene procesos, eventos y roles para poder realizar las siguientes actividades.

- Evaluar objetivamente los procesos realizados y los productos de trabajo contra descripciones de procesos, estándares y procedimientos aplicables.
- Identificar y documentar problemas de incumplimiento.
- Proporcionar retroalimentación al personal del proyecto y a los gerentes sobre los resultados de actividades de aseguramiento de la calidad.
- Asegurar que se aborden los problemas de incumplimiento.

3. Referencias

- Norma ISO 9001:2015 Sistema de Gestión de Calidad, Requisito 7.5.3.a Control de la información documentada.
- Norma ISO-9126 Evaluación de la calidad de software.
- Norma ISO-25000 SQuaRE Requisitos y evaluación de la calidad del software y del sistema.
- CMMI for Development, versión 1.3.

4. Términos y Definiciones

Calidad: grado en que las características inherentes a un objeto (producto, servicio, proceso, persona, organización, sistema o recurso) cumple con los requisitos.

Aseguramiento de la calidad: es un conjunto de actividades planificadas y sistemáticas que se aplican para verificar que los requisitos de calidad de un producto o servicio sean cumplidos.

Proceso: un proceso es un conjunto de actividades que están relacionadas entre sí y que transforman los elementos entrantes para obtener elementos de salida.

Procedimiento: es una descripción detallada de cómo se debe llevar a cabo un proceso.

Producto: es el resultado que se obtiene de un proceso, también se le define como un conjunto de características que puede ser tangible o intangible y que satisface ciertas necesidades.

Evaluación: proceso mediante el cual se intenta determinar el valor de un producto o proceso.

Estándar: punto de referencia para medir o valorar un proceso o producto.

Pruebas de software: son métodos para verificar si el producto de software coincide con los requerimientos esperados y para garantizar que el producto de software no tenga defectos.

No conformidades: es un incumplimiento de un requisito del sistema, sea este especificado o no.

Defecto: corresponde a un error, imperfecto o falla de una aplicación para computador que puede causar un resultado no deseado o incumplimiento de un requerimiento.

Desarrollo guiado por comportamiento: es un proceso de desarrollo ágil, que permite la colaboración entre los desarrolladores, los analistas de calidad, los gestores de proyecto y el negocio. Este se basa en la realización del código en base al comportamiento definido por el negocio y depurado por el equipo de aseguramiento de calidad.

5. Responsabilidades

5.1. **Encargado de departamento de calidad de software:** Responsable de todas las actividades relacionadas con la calidad de software (procedimientos y herramientas de control de calidad), así como también el control y mejora continua de la calidad.

5.2. **Director de Desarrollo y calidad:** Responsable de dirigir, coordinar y supervisar los departamentos de control de la calidad y desarrollo.

5.3. **Técnico de calidad:** Bajo la supervisión directa del encargado de calidad, realiza actividades de pruebas de software (caja negra, caja blanca y pruebas de estrés)

5.4. **Técnico de Desarrollo:** Bajo supervisión directa, realiza la codificación del producto en cuestión, así como las pruebas unitarias del software.

5.5. **Técnico de DevOps:** Bajo supervisión directa, realiza la codificación de la infraestructura que se despliega en la nube, así como también la configuración del CI/CD.

5.6. **Encargado de Desarrollo:** Responsable de todas las actividades relacionadas con el desarrollo de software, además vela para que la metodología scrum + XP sea parte siempre de los técnicos de desarrollo.

5.7. **Product Owner:** Responsable de todas las actividades relacionadas adquirir los requerimientos no funcionales y funcionales del cliente, para así transformarlos en historias de usuario para puedan ser trabajadas con el equipo de desarrollo

5.8. **Scrum Master:** Responsable de que todas las actividades de scrum se cumplan al pie de los lineamientos del scrum alliance.

6. Desarrollo

6.1. Evaluar objetivamente los procesos y los productos de trabajo

Se debe evaluar de manera objetiva si los procesos realizados y los productos de trabajo cumplen con las descripciones, los estándares y procedimientos correspondientes.

6.1.1. Evaluar objetivamente los procesos

El encargado del departamento de aseguramiento de la calidad es quien debe crear y modificar el proceso que se utiliza para llevar a cabo las actividades de aseguramiento de calidad.

Para evaluar objetivamente se establecieron los siguientes criterios:

- **Cuando o con qué frecuencia será evaluado el proceso:** el proceso debe ser evaluado cuatro veces al año, es decir, cada 3 meses, esto permitirá que se pueda determinar si las fases que se realizan están dando resultados positivos en un período de tiempo considerable.
- **Cuáles son los parámetros para llevar a cabo la evaluación:** los procesos deben ser evaluados en base a las descripciones, estándares y procedimientos aplicables.
- **Quién debe estar involucrado en la evaluación:** el encargado de calidad es quien debe evaluar si los técnicos de calidad están llevando el proceso a cabo y realiza la evaluación en base a los resultados de aplicación del mismo.

El resultado de la evaluación será representado por los siguientes colores:

 El proceso se encuentra funcionando de manera correcta, se pueden continuar las operaciones sin levantar no conformidades o hacer modificaciones.

 Se presentaron no conformidades menores, se puede seguir operando con el mismo proceso.

 Se presentaron no conformidades mayores, el proceso debe sujetarse a modificaciones.

Luego de la evaluación, las no conformidades encontradas deben ser documentadas en un acta que consta de tres partes:

- La evidencia de la no conformidad
- El requisito
- El enunciado de la no conformidad

El enunciado de la no conformidad, en este caso, debe:

- Ser autoexplicativo, es decir, que la no conformidad se entienda al leer el enunciado.
- No ambiguo

Luego de documentarlas, las no conformidades deben ser comentadas con el equipo y se llegan a soluciones que pueden resultar en modificaciones al proceso.

Para que la evaluación del proceso obtenga un resultado positivo debe promoverse una cultura que incentive la participación del personal en la identificación de las cuestiones de calidad, incluso si el personal no pertenece al área de aseguramiento de la calidad específicamente. Para cumplir con este objetivo, es necesario brindar capacitaciones que resalten la importancia de contar con un proceso de calidad y de cómo llevar a cabo las actividades de calidad, estas capacitaciones pueden ser charlas, cursos online, talleres, entre otros.

6.1.2. Evaluar objetivamente los productos de trabajo

Para evaluar objetivamente los productos de trabajo se debe seguir un proceso para el aseguramiento de la calidad, que cuenta con las siguientes fases:

Planeación: el personal del área de calidad de software debe revisar los requerimientos con el objetivo de identificar defectos tempranos, esta actividad se puede realizar en la reunión de sprint planning. Al revisar los requerimientos este tendrá el tiempo para definir una estrategia de prueba, seleccionar las herramientas correspondientes y estimar los tiempos de prueba.

Diseño: se deben crear casos de prueba que permitan explorar los distintos flujos de la aplicación, esto posibilitará que la cobertura de pruebas sea más amplia. Estos casos de prueba pueden ser documentados en una herramienta especializada o pueden ser escritos en una sintaxis basada en Desarrollo guiado por comportamiento (Behavior Driven Development o BDD), esta última recomendación debido a que la empresa realiza sus operaciones en base a una metodología ágil.

Ejecución: los ambientes de prueba deben ser configurados, se deben crear o extraer los datos a utilizar en las pruebas, se deben realizar las pruebas de software, las cuales dependen del producto que se vaya a probar, luego de las pruebas se deben reportar los resultados, es decir, si se encontraron no conformidades, si el producto no está listo o si

la prueba pasó sin problemas. Si se encuentran defectos, estos deben ser registrados, preferiblemente en Jira (Se define en el apartado 6.2.1).

Cierre: se debe evaluar cuáles tareas fueron completadas de manera satisfactoria, se debe determinar el futuro de los defectos y se debe crear un reporte de finalización.

Para aplicar el proceso detallado anteriormente se deben definir los siguientes criterios:

- **Qué producto será evaluado:** generalmente se refiere a la aplicación que se encuentra bajo pruebas.
- **Qué será evaluado dentro del producto:** se refiere específicamente al módulo o funcionalidad a probar.
- **Cómo se llevará a cabo la evaluación:** se refiere a la estrategia de pruebas a utilizar.
- **Quién debe estar involucrado en la evaluación:** en este caso, el técnico de calidad.

Dentro de la fase de ejecución, las pruebas de software son una de las actividades más utilizadas para evaluar objetivamente los productos de trabajo.

Dentro de estas pruebas se verifica que los productos cuenten con un conjunto de características. En la tabla 4.1 se muestran algunas características generales que todo producto de software debe tener:

Funcionalidad	Se enfoca en resultados generados en respuesta a ciertas acciones.
Usabilidad	Se debe medir qué tan fácil de utilizar y qué tan amigable es un producto de software. En este tipo de prueba el usuario navega por el producto y basado en su experiencia contesta preguntas como: ● ¿Es el sistema fácil de aprender? ● ¿Es el sistema funcional y agrega valor?

	● ¿Son los colores, íconos e imágenes usados, estéticamente satisfactorios? Cumple con objetivos como: ● Se necesita poca navegación para llegar a los lugares deseados. ● Las pantallas o páginas tienen un formato uniforme. ● No hay datos incorrectos o desactualizados. ● No hay enlaces rotos.
Eficiencia	Se debe determinar la velocidad, la capacidad de respuesta y la estabilidad de un sistema.
Mantenibilidad	Se debe evaluar si el sistema permite extenderse y modificarse.
Adaptabilidad	Se debe evaluar la capacidad de un sistema de ser transferido y utilizado en diferentes plataformas.

Tabla 4.1 - Características para evaluar un producto de software

6.2. Proporcionar una visión objetiva

Las no conformidades se deben seguir y comunicar de forma objetiva, y se debe asegurar su resolución.

6.2.1. Comunicar y resolver las no conformidades

Se define a las no conformidades como problemas identificados durante las evaluaciones, estas por lo general se tratan de una falta de adherencia a los estándares, descripciones de procesos y productos. Las tendencias de calidad se pueden determinar dependiendo del estado de las no conformidades.

Las no conformidades deben ser reportadas por los técnicos de calidad y deben ser resueltas por el equipo de desarrollo correspondiente. Las no conformidades del producto deben ser documentadas de la siguiente manera:

Se procederá a crear un ticket tipo bug en la herramienta Jira con el siguiente formato:

Título: debe resumir la problemática de tal manera que se pueda tener una idea de esta con solo leer el título.

Pasos para replicar: secuencia de acciones a realizar para encontrar el defecto.

Resultado actual: el comportamiento obtenido.

Resultado esperado: el comportamiento ideal.

Ambiente: plataforma en la que se encontró el defecto.

Imágenes: capturas o vídeos que ayuden a comprender mejor el problema.

Estas también deben ser analizadas con el objetivo de identificar tendencias de calidad a ser tratadas, se debe revisar periódicamente cuales quedan sin resolverse y qué acciones se pueden tomar para resolverlas.

6.2.2. Establecer los registros

Se deben establecer y mantener los registros de las actividades de aseguramiento de la calidad. Se deben registrar las actividades de aseguramiento de calidad del proceso y del producto con suficiente detalle, de forma que sean conocidos el estado y los resultados.

En el siguiente punto se detallan los registros a tomar en cuenta en el nuevo proceso de evaluación de procesos y de productos.

7. Registros

- **Requerimientos:** historias de usuario que cuentan con una descripción y criterios de aceptación, estas cuentan con el formato predeterminado de Jira.
- **Defectos:** artefacto que cuenta con pasos para replicar, resultados actuales y esperados, ambiente, evidencias, severidad del defecto y prioridad, estos serán creados en Jira.
- **Reporte de finalización de pruebas:** reporte que indique que las pruebas han sido terminadas.
- **Acta de no conformidad:** documento que describe las no conformidades encontradas durante las evaluaciones.

8. Historial de cambios

Cambios y Revisiones			
Versión	**Fecha**	**Descripción**	**Realizado Por**
1	23/Oct/2020	Propuesta de mejoras al proceso de calidad de software actual de la empresa Wepsys S.R.L.	Carla Gómez Luis Javier

Resumen del capítulo IV

En este capítulo se describió la propuesta de mejora al proceso de aseguramiento de calidad de software de la empresa Wepsys, siguiendo los lineamientos y las mejores prácticas recomendadas dentro del área clave de proceso PPQA perteneciente a CMMI, el cual es un modelo reconocido y utilizado internacionalmente.

Dentro de esta propuesta se definieron criterios que permiten la evaluación de los procesos y los productos, así como la proporción de una visión objetiva en cuanto a las no conformidades y a los registros, esto con el fin de satisfacer las necesidades de la empresa en cuanto al proceso de calidad de software.

CONCLUSIÓN

El objetivo de este trabajo de grado consistió en determinar el nivel de madurez del proceso de calidad de software de la factoría de software Wepsys S.R.L. e implementar mejoras en base a los resultados. La propuesta solventará la problemática de que se lleven a cabo actividades de aseguramiento de la calidad de manera incompleta y que por tal razón se encuentren defectos en etapas tardías.

Para lograr este objetivo, en primera instancia se describieron los conceptos que sirvieron como base para sustentar este proyecto. Este punto se relaciona de manera directa con el **1er objetivo específico** que consistió en **describir la importancia del proceso de calidad dentro del ciclo de vida del desarrollo de software** y que corresponde al capítulo I de esta investigación, el cual es el marco teórico y del mismo se puede concluir que:

- La calidad es el objetivo de todas las prácticas que se han implementado en la historia de la ingeniería de software, incluyendo a esta última, la cual fue creada para dar resolución a problemas de calidad.

- El aseguramiento de la calidad es una actividad de gran relevancia dentro del ciclo de vida del desarrollo de software y no solo se debe realizar luego de la construcción del software, sino que, debe empezar desde los requerimientos.

- CMMI de igual forma tiene como objetivo que los procesos y productos se encuentren en un nivel alto de calidad.

Luego se definió la **metodología de investigación a utilizar** para poder obtener los datos necesarios y realizar el análisis del proceso de calidad de software, esto corresponde al **2do objetivo específico** y fue desarrollado en el capítulo II, en el cual se llegó a las siguientes conclusiones:

- Se utilizó el método cuantitativo para obtener los datos que prueben la validez de la investigación, ya que este define la técnica de recolección de datos a utilizar, que en este caso fue una encuesta.

- Se determinó el tamaño de la muestra utilizando el método aleatorio simple para poblaciones finitas, pero, debido a que la población estaba compuesta de los colaboradores de las áreas desarrollo y gestión de proyectos, los cuales son 28 personas en total, la fórmula aplicada determinó que la muestra no varió con relación a la población, por tanto, se realizó la encuesta al 100% de la población.

- Los resultados de la encuesta fueron representados mediante la utilización de gráficas.

- Se utilizó el método inductivo para obtener conclusiones acerca del proceso de calidad de software actual con base en los resultados obtenidos en la encuesta.

Al contar con la descripción de la metodología de investigación, se desarrolló el capítulo III, el cual tiene como propósito dar respuesta al **3er objetivo específico**, que consistió en **analizar el estado actual y las oportunidades de mejora del proceso de calidad de software de Wepsys**. La primera tarea de este capítulo fue describir las actividades principales de la empresa, junto con su misión, visión, valores y estructura organizacional, también se realizó un análisis FODA, que permitió comparar las fortalezas, oportunidades, debilidades y amenazas de la empresa. Dentro de las debilidades del análisis FODA se resaltó que el proceso de calidad no se encuentra bien definido.

Con el fin de validar la veracidad de la debilidad del proceso de calidad de software, se procedió a describir el ciclo de vida del desarrollo software, con el objetivo de analizar el papel que juega el aseguramiento de la calidad en el mismo y se utilizó el instrumento de recolección de datos propuesto en el capítulo anterior, es decir, la encuesta, la cual confirmó los siguientes resultados:

- El 61% de los empleados de las áreas de desarrollo y proyectos entiende que las prácticas recomendadas por el Modelo de Mejora de Capacidad Integrado (CMMI) ayudarían a mejorar el proceso de calidad de software.

- El 57% de los encuestados asegura que no se han adoptado principios de ninguna de las escuelas de calidad.

- Solo el 39% está de acuerdo con que se sigue una política organizacional escrita y aprobada para la gestión de requerimientos de sistema. El 61% restante, no está

de acuerdo o seguro de si se sigue una política organizacional para la gestión de requerimientos.

- Tan solo el 29% de los encuestados no está de acuerdo con que se dejan asuntos incumplidos que no se resuelven en los proyectos, mientras que un 32% está de acuerdo en que sí se quedan asuntos incumplidos sin resolver y un 39% sostiene una posición neutral al respecto.

- Solo el 50% de los empleados encuestados considera que se sigue una política organizacional escrita y aprobada por la máxima autoridad para la implementación del aseguramiento de calidad, mientras que el 50% restante no está de acuerdo o no está seguro.

- Solo un 46% está de acuerdo o totalmente de acuerdo con que las actividades de calidad cuentan con los recursos adecuados, mientras que un 29% no está ni de acuerdo, ni en desacuerdo y un 25% se encuentra en desacuerdo o totalmente en desacuerdo.

- Un 53% no está de acuerdo o seguro de si las actividades de aseguramiento de la calidad se revisan periódicamente.

- Se realizó una pregunta abierta para que los colaboradores realicen propuestas de mejora para el proceso de calidad y la mayoría coincidió en que se necesitan capacitaciones para el personal, que se necesita crear un proceso escrito y que se deben crear artefactos como defectos.

Luego de realizar el análisis del proceso actual e identificar sus debilidades, fue desarrollado el capítulo IV, en el cual procedimos a **proponer mejoras al proceso de calidad de software con base en el área de proceso de Aseguramiento de la calidad de procesos y de productos (PPQA) del nivel 2 de CMMI**, el cual fue desarrollado por el Instituto de Ingeniería de Software de la Universidad Carnegie Mellon, esto permitió que se lograra el **4to objetivo específico.**

En este capítulo se describió el área clave de proceso de Aseguramiento de la calidad de procesos y de productos (PPQA) y a desarrollar una guía para la evaluación de los procesos y productos. Dentro de la evaluación de los procesos se definieron criterios como:

- El encargado de llevar a cabo la evaluación.
- Quién debe dar mantenimiento al proceso.
- Con qué frecuencia se evaluará el proceso.
- Recomendaciones para que las evaluaciones del proceso sean exitosas.

Dentro de la evaluación de los productos, se determinó que la manera idónea de evaluar los productos es mediante la realización de pruebas de software, partiendo de esto se creó un proceso que consta con las siguientes fases:

- Planeación
- Diseño
- Ejecución
- Cierre

Por último, es importante que todas las no conformidades sean comunicadas y debidamente documentadas, para resolver las problemáticas actuales y prevenir no conformidades futuras.

Este nuevo proceso se encarga de tomar en cuenta las variables necesarias para que los productos construidos cuenten con un alto nivel de calidad y así las necesidades de los clientes puedan ser satisfechas.

RECOMENDACIONES

Tomando en consideración que la necesidad de tener un control de calidad de software es de suma importancia a la hora de amedrentar costos y gastos en mantenimiento, esto sumado a que la calidad juega un papel de mucha importancia a la hora de entregar un producto o servicio se recomienda lo siguiente:

- El proceso de control de calidad deberá estar incluido en el proceso de desarrollo de software.
- Realizar campañas de educación a los miembros del departamento de desarrollo para que conozcan más a fondo las metodologías que se utilizan en la empresa.
- Impartir capacitaciones de International Software Testing Qualifications Board (ISTQB) a los empleados para el aprendizaje de herramientas básicas de control de calidad de software.
- Incluir de manera formal en la empresa los procesos de calidad pertinentes para cada proyecto.
- Realizar evaluaciones trimestrales para verificar que el grado de calidad de cada proyecto o producto realizado continúa siendo el estándar de la empresa.
- Documentar el proceso de calidad de software para que los empleados del área puedan entender los lineamientos de la empresa.
- Analizar los defectos comunes para que se puedan identificar patrones o causas comunes para que estos puedan ser prevenidos en el futuro.

REFERENCIAS BIBLIOGRÁFICAS

1. Carrizo, D., & Alfaro, A. (2018). Método de aseguramiento de la calidad en una metodología. *Ingeniare*.

2. Chaudhary, M., & Abhishek, C. (2017). CMMI for Development: Implementation Guide. Apress.

3. Chopra, R. (2018). *Software Quality Assurance: A self- teaching introduction.* Dulles, Virginia, United States of America: Mercury Learning and Information.

4. Claude Y. Laporte. (2018). *Software Quality.* IEEE PRESS.

5. Córdoba, M., & Monsalve, C. (s.f.). *Tipos de investigación: Predictiva, proyectiva, interactiva, confirmatoria y evaluativa.* Universidad Pedagógica Nacional, Bogotá. Obtenido de http://2633518-0.web-hosting.es/blog/didact_mate/9.Tipos%20de%20Investigación.%20Predictiva%2C%20Proyectiva%2C%20Interactiva%2C%20Confirmatoria%20y%20Evaluativa.pdf

6. Dooley, J. (2017). *Software Development, Design and Coding.* Galesburg, Illinois, United States of America: Apress.

7. Dijkstra, E. W. (n.d.). Notes on Structures Programming. In *Notes on Structures Programming* (pp. 15-64).

8. Establecimientos, R. N. (2015). *Estadísticas de Empresas TIC.* Santo Domingo, República Dominicana.

9. IEEE. (n.d.). Guide to the software Engineering Body of Knowledge. In *SWEBOK V3.*

10. Evans Data Corporation. (2019). *Worldwide Professional Developer Population of 24 Million Projected to Grow amid Shifting Geographical Concentrations.* Santa Cruz,California. Retrieved septiembre 18, 2020, from https://evansdata.com/press/viewRelease.php?pressID=278

11. International Organization for Standardization. (2015). ISO 9001.

12. International Software Testing Qualification Board. (2018). Syllabus - Certified Software Tester Foundation Level.

13. Laporte, C., & Alain, A. (2018). *Software Quality Assurance.* Hoboken, New Jersey, United States of America: IEEE Computer Society, Inc.

14. MiPymes, O. (2020). *Conoce cómo las MiPymes del sector TIC prosperan en la nueva era digital.* Santo Domingo, República Dominicana.

15. Mitra, A. (2016). *Fundamentals of quality control and improvement* (4th ed.). Hoboken, New Jersey, United States of America: Wiley.

16. Pardo, C., Hurtado, J. A. & Collazos, C. A. (2018). *Mejora de procesos de software ágil con Agile Process.*

17. Pressman, R., & Maxim, B. (2015). *Software Engineering: A practitioner's approach* (8th ed.). New York City, New York, United States of America: McGraw-Hill Education.

18. Software Engineering Institute. (2010). CMMI para Desarrollo (1.3 ed.). Editorial Universitaria Ramón Areces.

19. Sommerville, I. (2016). *Software Engineering* (10th ed.). Harlow, Essex, England: Pearson Education Limited.

20. Zumba, P. & Cecibel, L. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. *INNOVA Research Journal*.

ANEXOS

Anexo 1. Encuesta

DECANATO DE INGENIERÍA E INFORMÁTICA

ESCUELA DE INFORMÁTICA

ANÁLISIS DEL ESTADO ACTUAL DEL PROCESO DE CALIDAD DE SOFTWARE DE LA FACTORÍA DE SOFTWARE WEPSYS S.R.L., SANTO DOMINGO, REP. DOM.

ENCUESTA

Esta encuesta tiene por objetivo conocer la percepción de los trabajadores de Wepsys S.R.L. en relación con el proceso de calidad de software, con el fin de conocer su estado actual y proponer mejoras. Contiene 24 preguntas, solo le tomará unos minutos en responder. Su sinceridad nos ayudará a valorar con precisión el estado actual del proceso de calidad de software.

¡Muchas Gracias por su colaboración!

INFORMACIÓN GENERAL ACERCA DE WEPSYS S.R.L.

Cantidad de Empleados: _____

Puestos: _____

_____

_____

_____

Aplicaciones que comercializan: _____

_____

_____

_____

Mercado al que está dirigida:

□ Nacional
□ Internacional
□ Ambos

Tipo de Producto:

□ Aplicaciones Web
□ Aplicaciones móviles
□ Todas las anteriores
□ Otras Especifique: _____

CUESTIONARIO GENERAL

1. ¿Cuál es su puesto de trabajo actualmente?

☐ Director a nivel de la organización
☐ Jefe de proyectos
☐ Líder de equipo
☐ Desarrollador
☐ Área de calidad
☐ Otro

2. ¿Realiza otra actividad aparte de las mencionadas anteriormente?

☐ Sí Especifique: _____
☐ No

3. Experiencia

En esta organización: ____ años

En el área de software: ____ años

4. ¿Sigue su empresa algún modelo o metodología nacional o internacional de desarrollo de software?

□ Sí
□ No

5. ¿En cuáles de estos modelos de desarrollo de software se basa su empresa?

□ CMMI
□ ISO
□ SPICE
□ Otros Especifique: _____
□ Ninguno

6. ¿Posee su empresa una certificación en algunos de los modelos de desarrollo de software mencionado anteriormente?

□ Sí
□ No

7. ¿Tiene su empresa conocimientos de algunos de los modelos de desarrollo de software mencionado anteriormente?

□ Sí Especifique: _____
□ No

8. ¿Cuál es el nivel de aplicación del modelo de desarrollo de software que utiliza su empresa? (Opcional)

□ Alto
□ Medio
□ Bajo

9. ¿Ha adoptado su empresa algunos principios de las escuelas de la calidad con la finalidad de mejorar el producto que entregan al cliente?

□ Sí Especifique cuales: _____
□ No

10. ¿En cuales lenguajes de programación, sistemas operativos y base de datos están desarrollados las aplicaciones que comercializa su empresa?

□ Sistema Operativo _____
□ Base de Datos _____
□ Lenguaje de programación _____

EVALUACIÓN DEL NIVEL DE SATISFACCIÓN

Marque con una equis (✕) la respuesta que considere, tomando en cuenta que:

No.	Significados	
1	**Totalmente en desacuerdo**	**Muy malo**
2	**En desacuerdo**	**Malo**
3	**Ni de acuerdo, ni en desacuerdo**	**Regular**
4	**De acuerdo**	**Muy bueno**
5	**Totalmente de acuerdo**	**Excelente**

Preguntas		Nivel de conformidad				
		1	**2**	**3**	**4**	**5**
1	¿Los Requerimientos de Sistema asignados al software son usados para establecer una Línea Base para el uso de la Ingeniería y Gestión de Software?					
2	¿Cuándo los Requerimientos de Sistema cambian, se hacen los ajustes necesarios a los planes, productos de trabajo de software y otras actividades relacionadas?					
3	¿El proyecto sigue una política organizacional escrita para la gestión de los Requerimientos del Sistemas asignados al software?					
4	¿Se sigue una política organizacional escrita que rija el planeamiento del proyecto de software?					
5	¿El proyecto sigue una política organizacional escrita tanto para el seguimiento como para el control de las actividades de desarrollo del software?					
6	¿Se planifican las actividades de Aseguramiento de la calidad de software?					
7	¿El aseguramiento de la calidad suministra una verificación objetiva de que las actividades y productos de software se corresponden con los procedimientos, estándares aplicables y con los requerimientos?					
8	¿Los resultados de las revisiones de aseguramiento de la calidad, se le suministran a los individuos y grupos afectados (es decir,					

	aquellos quienes realizan el trabajo y aquellos responsables del trabajo)?					
9	¿Quedan asuntos incumplidos que no se resuelven en el proyecto de software (ej. Desviaciones de los estándares aplicables)?					
10	¿El proyecto sigue una política organizacional escrita y aprobada por la máxima autoridad para la implementación del aseguramiento de la calidad?					
11	¿Las actividades de aseguramiento de la calidad cuentan con los recursos adecuados (¿ej. fondos y personas entrenadas, personas que dirijan y se responsabilice con la actividad?					
12	¿Se usan mediciones para determinar el costo y el estado del cronograma de las actividades realizadas por aseguramiento de la calidad (ej. Trabajo terminado, esfuerzos y fondos gastados comprados por el plan)?					
13	¿Las actividades de aseguramiento de la calidad son revisadas periódicamente?					

Anexo 2. Anteproyecto de trabajo de grado

UNAPEC
DEPARTAMENTO DE CIENCIAS SOCIALES

A : ESCUELA DE INFORMÁTICA.

Asunto: **REMISIÓN ANTEPROYECTO DE TRABAJO DE GRADO.**

Tema : "Análisis del estado actual del proceso de Calidad de Software de la factoría de software Wepsys S.R.L., Santo Domingo, Rep. Dom."

Sustentado por:	**Br. Carla Milenis Gómez Pérez**	**2017-0012**
	Br. Luis José Javier Tatis	**2017-0259**

Resultado de la evaluación: Aprobado: ____X____ Fecha: 20/07/2020

Devuelto para corrección: _____ Fecha: _____

_____
Dr. Andrés L. Mateo
Decano de Estudios Generales

AM/ra. 20/07/2020.

Decanato de ingeniería e informática

Escuela de Informática

Anteproyecto de Trabajo de Grado para optar por el título de Ingeniero de Software

"Análisis del estado actual del proceso de Calidad de Software de la factoría de software Wepsys S.R.L., Santo Domingo, Rep. Dom."

Sustentantes

Carla Milenis Gómez Pérez 2017-0012
Luis José Javier Tatis 2017-0259

Asesor

Ing. Luis Gabriel Núñez

Santo Domingo, Distrito Nacional,
República Dominicana,
Julio 2020

Introducción

Cuando se empezaron a construir los primeros productos de software, no se habían desarrollado los procesos necesarios para llevar a cabo esta tarea sin que ocurran problemas mayores, a esto se le llamó “La crisis de software”. A raíz de esta crisis se creó la Ingeniería de Software, la cual se encargó de estructurar la manera en que se construían los productos, también ayudó a minimizar riesgos y a reducir costos. A partir de esto también se desarrolló lo que hoy conocemos como ciclo de vida de un software.

Con la implementación de la Ingeniería de Software, muchos errores fueron reducidos, pero, los incidentes seguían pasando porque los mismos desarrolladores eran los encargados de realizar pruebas a lo que construían. A lo largo de la historia se registraron eventos que se convirtieron en pérdidas millonarias, algunos de estos fueron cohetes que explotaron por errores de software, radiación excesiva por parte de máquinas de quimioterapia, entre otras catástrofes que resultaron en pérdidas de vida humana. Esas fueron algunas de las razones por la cual surgió un área dedicada al aseguramiento de la calidad de software.

La calidad de software hace énfasis en que procesos como la especificación de requerimientos, el análisis, el diseño y el producto final cumplan con las necesidades de los usuarios y que tengan un funcionamiento correcto.

No cabe duda de que el aseguramiento de la calidad de software se ha convertido en el proceso clave que se utiliza para evaluar si el producto desarrollado cumple con los estándares establecidos, de hecho, algunos autores se refieren a las pruebas de software (una de las actividades englobadas en el proceso de calidad) como la única actividad que se utiliza para asegurar la calidad de los productos de software.

Las actividades de calidad eran menospreciadas por tardar mucho tiempo y realizarse solo una vez en el ciclo de vida del software, no fue hasta hace un poco más de 20 años que las metodologías de desarrollo comenzaron a modificarse y a mejorar. Con las mejoras hechas el aseguramiento de la calidad comenzó a incluirse no solo en una fase, sino, que todas las fases contenían una actividad de prueba para ayudar a detectar defectos de manera temprana, lo cual supone una reducción del tiempo y los costos y lo más importante, la satisfacción del cliente.

Con los cambios, la calidad de los productos aumentó y a medida que se iban modificando las metodologías, estas se orientaron más a la calidad, desarrollando así métodos ágiles, para el desarrollo de software y así las actividades de calidad pasaron de ser procesos secundarios a ser parte del objetivo de cualquier software en construcción.

Es por esa razón que es imprescindible para cualquier empresa que se encargue de brindar soluciones de software, el tener un proceso de calidad que esté bien estructurado, que se base en un estándar definido y que cumpla con su objetivo, en pocas palabras, que sea maduro.

Uno de los indicadores que se pueden utilizar para saber si un proceso de calidad de software es maduro, es el área de proceso PPQA (Aseguramiento de la calidad de procesos y productos) perteneciente al nivel 2 del modelo de mejora CMMI (Integración de Modelos de Madurez de Capacidades).

Justificación

Cada día el sector de software en República Dominicana crece, tanto a nivel de profesionales que se gradúan de las universidades en Ingeniería de Software o Sistemas, como de factorías que se dedican al desarrollo de software como servicio. Desde hace más de una década el área de Calidad de Software, la cual es la encargada de realizar pruebas a los productos que se desarrollan, se ha consolidado como una práctica formal en el país, sin embargo, aún hay empresas que no han implementado departamentos de calidad y si lo han hecho, no cuentan con un proceso de calidad eficaz.

El proceso de calidad de software consta de actividades que se ocupan de asegurar que se cumplan los requisitos funcionales y no funcionales de un producto y que el cliente se encuentre satisfecho con los resultados.

Es de gran importancia que las empresas que se dedican a la comercialización de software cuenten con un proceso de calidad con un alto nivel de madurez puesto que esto trae una gran cantidad de beneficios, tales como, entregar un producto de calidad a los clientes con la menor cantidad de defectos posibles, reducir costos operativos, costos de reparación de defectos, costos de mantenimiento, evitar que se extienda el tiempo de duración de un proyecto y que se deban aumentar los recursos destinados a este, aumentar la confianza de los clientes, quienes tienen la posibilidad de recomendar los servicios de la empresa, mejorar las experiencias de los usuarios, entre otros.

Las recomendaciones forman una gran parte de la cultura de los negocios, de hecho, al momento de solicitar un servicio, la mayoría de los consumidores lo solicita a través de empresas recomendadas.

Por esa razón es importante mantener la reputación de la empresa intacta y uno de los medios para lograr este objetivo es tener un proceso de calidad de software que esté desarrollado de manera correcta.

Al analizar todos los beneficios que se obtienen al tener un proceso de calidad de software con un nivel de madurez alto, es evidente que es una necesidad para las empresas que se dedican a la comercialización de software.

Para poder obtener todos los beneficios mencionados lo recomendable empezar realizando un análisis del estado actual del proceso de calidad de software en las empresas que desean obtener resultados favorables, en esta ocasión, Wepsys S.R.L., esto con el

objetivo de determinar cuál es el nivel de madurez del proceso, poder comparar los resultados con las características de un proceso estandarizado y a partir de los hallazgos poder implementar técnicas que ayuden a mejorar el proceso de calidad de la empresa.

Planteamiento del problema

En la última década, el área de pruebas de software ha tenido un auge significativo en el desarrollo de soluciones tecnológicas y las pruebas han pasado de ser una actividad informal a un área con procesos definidos, donde muchas empresas se han embarcado en la tendencia de contar con un departamento de Calidad de Software.

Wepsys S.R.L. es una factoría de software fundada en el 2007, desde entonces se dedica al desarrollo de software de manera ininterrumpida, colaborando con varios clientes tanto en el sector privado como público. Ha sido reconocido por Microsoft como Gold Partner en Application Builder.

Con el crecimiento obtenido en en los últimos 5 años, Wepsys S.R.L. ha implementado diferentes procesos del manejo del ciclo de vida ALM (Application Lifecycle Management), del cual las pruebas de software ocupan un importante lugar, porque es lo que asegura la calidad de sus productos.

Debido a esto la factoría decidió tomar la iniciativa en cuanto a la creación del departamento de calidad. Este departamento fue inaugurado recientemente y cuenta con un proceso de calidad de software relativamente nuevo. A esto se le suma que los colaboradores asignados al departamento cuentan con poca experiencia en el área del aseguramiento de la calidad, haciendo que el proceso de pruebas se sienta lento y no entregue los resultados esperados. Este proceso nunca ha sido evaluado, por lo que no se ha determinado cuales son las causas de que el proceso no sea completamente funcional.

Por esta razón es necesario realizar un análisis del estado en el que se encuentra el proceso de calidad de software en la factoría Wepsys y de esa manera poder detectar los errores del proceso y contestar por qué no está funcionando de manera óptima, también determinar cuáles mejoras se deben aplicar para que se cumpla el objetivo de brindar calidad a los clientes y aumentar la confianza.

Formulación del problema

¿Cuáles son las oportunidades de mejora en el proceso de calidad de software de la empresa Wepsys?

Sistematización del problema

¿Por qué es importante tener un proceso de calidad dentro del ciclo de vida del desarrollo de software?

¿Qué métodos de investigación se utilizarán para evaluar el proceso de calidad de software existente en la factoría de software Wepsys S.R.L.?

¿Cuál es el estado actual del proceso de calidad de software de Wepsys S.R.L.?

¿Cuáles son las propuestas correspondientes para mejorar el proceso de calidad de software de Wepsys S.R.L.?

Objetivo general

Determinar el nivel de madurez del proceso de calidad de software de la factoría de software Wepsys S.R.L e implementar mejoras en base a los resultados.

Objetivos específicos

Describir la importancia del proceso de calidad dentro del ciclo de vida de desarrollo software.

Determinar los métodos de investigación a utilizar para evaluar el estado actual del proceso de calidad de software de la factoría de software Wepsys S.R.L.

Analizar el estado actual y las oportunidades de mejora del proceso de calidad de software que se utiliza en la factoría de software Wepsys S.R.L.

Proponer mejoras basadas en el área de proceso de Aseguramiento de la Calidad del Proceso y del Producto (PPQA) del nivel 2 del Modelo de Capacidad de Madurez Integrado (CMMI).

Marco teórico referencial

1.1.1 Ingeniería de Software

El diccionario de la RAE (2019) define **software** como "Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora." (RAE). De acuerdo también con webster's New Intercollegiate Dictionary "Software is the entire set of programs, procedures and related documentation associated with a system and especially a computer system." (Webster's New Intercollegiate Dictionary, 1981).

La **ingeniería** son el conjunto de técnicas y conocimientos que junto a la invención aprovechan la materia prima de un área para la actividad industrial. Según el New Intercollegiate Webster's Dictionary la define de la siguiente forma "the application of science and mathematics by which the properties of matter and the sources of energy in nature are made useful to man in structures, machines, products, systems and processes" (Webster's New Intercollegiate Dictionary, 1981).

Estas definiciones indican el camino para definir **ingeniería de software**, el cual sería el establecimiento de métodos científicos para el diseño y la construcción de software para poder documentar, operar y mantener el desarrollo de los mismo. Autores como Boehm (1972) lo definen como "The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate and maintain them" (Boehm, 1976). También como Bauer que dio una definición un poco más temprano de lo que para él es ingeniería de software" The establishment and use of sound engineering principles (methods) in order to obtain economically software that is reliable and works on real machines." (Bauer, 1972).

1.1.2 Antecedentes

Para hablar de la historia de la ingeniería de software, tenemos que remontarnos a 1964 cuando IBM combinó su sistema de computadora IBM S/360 con características tanto científicas como de negocios, este conjunto de máquinas hizo que las personas trataran de desarrollar softwares grandes y complejo generando así la necesidad de tener una disciplina que maneje el desarrollo de software. En 1969 en la conferencia de NATO sobre software engineering fue la primera vez que se utilizó el término de ingeniería de software, el cual fue a acreditado a Buer. Personajes como P. Nauer, J.N Buxton y

Dijkstra también realizaron grandes aportes para que la ingeniería de software fuera creciendo en esos años.

La primera conferencia internacional sobre ingeniería de software fue en 1973 por el IEEE. En los años siguientes Boehm realizó un famoso documento titulado Software Engineering (Boehm, Software Engineering, 1976) enmarcado claramente cuál era el marco de trabajo de la ingeniería de software. Ya en el 1975 F.Brooks, quien fue director del proyecto de desarrollo del sistema operativo de IBM 360 por un periodo de 10 años escribió un libro muy famoso en la ingeniería de software como lo es "The Mythical Man-Month" donde se mostraron muchísimos problemas con respecto al desarrollo de grandes softwares en un ambiente donde trabajan numerosas cantidades de personas.

Durante los años 1960 y 1970, creció una gran crisis con respecto al software, una crisis que tenía síntomas que incluso duraron hasta la actualidad. Uno de estos síntomas Boehm (1981) lo indica como que el software está incrementando muchísimo más su costo con relación al hardware.

1.1.3 Ciclo de vida del software

Es un esquema o esqueleto que se aplica a un producto de software. A través de los años se han desarrollado diferentes modelos a seguir para establecer un proceso de software efectivo. Cada uno de estos modelos utilizan unas actividades diferentes durante el proceso para poder llegar así al producto final.

Entre las actividades que deben realizarse durante el ciclo de vida de un proyecto de software se encuentra la planificación, implementación, pruebas, documentación, despliegue y mantenimiento.

La **planificación** consiste en adquirir los requerimientos que el cliente tiene en su cabeza para analizarlos y transformarlos en funciones y características que el software debería cumplir.

Por otra parte**, la implementación** es donde los desarrolladores escriben el código necesario para cumplir los requerimientos y demandas del cliente.

Las **pruebas** según (Singh, 2011) son un importante aspecto del ciclo de vida del desarrollo de software. Es donde se prueba lo desarrollado antes de que se use en el ambiente de producción.

La **documentación** ayuda a facilitar la mejora y mantenimiento del software durante su desarrollo y puesta en producción. Esta documentación puede incluir tanto diseño interno del software, como información de ambiente del mismo o incluso documentación de interfaces públicas como privadas.

El **despliegue** comienza cuando el código está listo para pasar a un ambiente de producción, pero no sin antes pasar por todas las pruebas concernientes a su liberación.

El mantenimiento o mejora de software se centra en resolver los nuevos problemas que han sido desplegados, aquí se pueden agregar incluso más piezas de código que el diseño original todo para solucionar el problema o incrementar las funciones del software.

1.1.4 Metodología de desarrollo de software

Para desarrollar un producto de software es necesario seguir un conjunto de pasos estructurados que permitan convertir los requisitos de los clientes en el producto deseado, que se realice a tiempo y que sea funcional, es a este proceso completo que se le conoce como metodologías o modelos de desarrollo de software.

Metodologías tradicionales

Según Awad (2005), durante la década del 1960, lo que se utilizaba era el método de "programa y arregla", pero, no era un método efectivo, por lo que Winston Royce en el 1970 propuso la metodología conocida como cascada.

Este modelo se basa en una serie de fases que tienen definida una serie de actividades y entregables que solo se pueden completar antes de que la fase siguiente comience.

La **metodología de Desarrollo en Cascada** se asume cuando el objetivo es entregar un producto completo una vez que la secuencia haya concluido (Pressman, 2002).

Según Sommerville (2011), este modelo solo se debe utilizar cuando los requerimientos estén bien definidos, se entiendan correctamente y no haya posibilidad de cambios radicales en el software a construir.

Luego de que fue adoptada, se hizo notable una característica que perjudicaba la calidad del producto construido, el modelo en cascada no permitía que el sistema evolucionara y por esa razón se decidió modificar la tradicional metodología lineal para incorporar un método secuencial que permitiera construir y entregar prototipos para reducir la cantidad de fallos. Sin embargo, el **Modelo Lineal Secuencia**l todavía quedaba por debajo de ser un modelo evolutivo.

Pressman (2002) expone que es por esta razón que se crea el **Modelo Incremental**, el cual combina la metodología lineal secuencial con una filosofía interactiva que utiliza una forma escalonada mientras avanza.

Luego de años de utilizar el modelo cascada e incremental Boehm expone los problemas de estos métodos y propone el Modelo en Espiral. El **Modelo en Espiral** integra lo mejor de los anteriores y trata de iterar las fases que fueron conocidas en los lineales secuenciales. Básicamente cuando termina un ciclo de desarrollo, inmediatamente empieza otro, siendo en su máxima expresión un modelo orientado a la evolución del producto (Boehm, 1988).

A pesar de que el Modelo en Espiral tiene un enfoque evolutivo, otras metodologías tradicionales surgieron, una de estas es el **Proceso Unificado**, que en su máximo esplendor es descrita como una metodología iterativa e incremental que hace énfasis en la documentación y en el uso de artefactos de software como los distintos diagramas UML, los casos de uso, entre otros

Metodologías ágiles

Los métodos ágiles combinan los modelos iterativos e incrementales con la practicidad de liberar el producto por entregables, en lugar de un producto completo una sola vez. Fueron desarrollados con la mentalidad de hacer el proceso de desarrollo más rápido o como su nombre describe, más ágil y de transformar el desarrollo de software en un proceso totalmente evolutivo orientado a la calidad.

Algunas metodologías ágiles son Extreme Programming (XP), Scrum y Kanban.

Extreme Programming: Según Beck (2000), XP es una metodología eficiente, flexible y de bajo riesgo, que hace que desarrollar software sea divertido. Es concreta y se basa en la retroalimentación de los resultados obtenidos en ciclos de desarrollo pequeños.

Tiene como característica la flexibilidad en el calendario de implementación, esto como respuesta a las necesidades cambiantes del negocio.

XP se caracteriza por ciclos cortos, retroalimentación continua y comunicación. El objetivo es responder a las necesidades de cambios de ambiente y necesidades del usuario (Awad, 2005).

Scrum: Es una forma de gestionar proyectos de manera ágil y flexible, que tiene como idea principal la creación de ciclos o sprints, donde se eligen una cantidad de funcionalidades a construir, las iteraciones se gestionan a través de reuniones diarias (Trigas, 2012).

Según (Awad, 2005), Scrum es un proceso iterativo e incremental que se concentra en cómo cada miembro de un equipo debe funcionar para lograr flexibilidad en un sistema que se encuentra en un ambiente cambiante. Scrum no requiere de metodologías definidas, pero sí requiere de prácticas de gestión para que el objetivo sea logrado.

Comparte varias características con XP, una de estas es la comunicación y participación del cliente (Laínez, 2014).

Kanban: En esta metodología ágil se divide el trabajo por bloques y utiliza un tablero donde se ponen los elementos a trabajar. Mide el tiempo de trabajo y trata de optimizarlo para que este tiempo sea lo más corto posible. Tiene como objetivo desplegar desde que se termina una funcionalidad, a diferencia de otras tecnologías que despliega un conjunto de cambios juntos (Kniberg & Skarin, 2014).

1.1.5 Desarrollo y Calidad de Software en República Dominicana

En República Dominicana la industria del software se puede describir como incipiente o poco desarrollada. Este sector surgió con un enfoque de construir sistemas administrativos y financieros que ayuden a eficientar los procesos de esta índole. De hecho, el 16% de sistemas que se realizan en el país son de cara al sector público, un 15% para el sector bancario y un 13% para el manejo de inventarios. Esto indica que tan poco desarrollada está el área de software, dónde en otros países ya se está trabajando en conjunto con la Inteligencia artificial para dar solución a problemas de carácter social.

Según un análisis presentado por el Observatorio MiPyMEs (2020), auspiciado por el Ministerio de Industria, Comercio y MiPyMEs, el programa República Digital y el

Instituto Tecnológico de Santo Domingo (INTEC), en el país, para el año 2017 había un total de 2,253 MiPyMEs TIC, del cual el 58.4% están concentradas en el Distrito Nacional, mientras que un 15.8% en Santo Domingo y un 9.5% en Santiago.

Sin embargo, de acuerdo con el Registro Nacional de Establecimientos (2014), es una meta para el país trabajar en pos del desarrollo de este sector debido a que es una actividad de alto impacto económico, con una alta tasa de innovación, competitividad y productividad, convirtiéndose en una fuente de reducción de pobreza. Este sector ha creado aproximadamente 39,000 empleos.

De todas MiPyMEs TIC que existen en República Dominicana, la mayoría de estas no cuentan con un proceso de calidad separado del equipo de desarrollo de software, lo cual incrementa exponencialmente, el riesgo a comercializar productos defectuosos, a invertir más dinero del presupuestado y a tardar más para operar en un ambiente de producción.

En el 2019 se creó el primer Instituto Dominicano de Calidad de Software, también el programa Becasoft incluyó en su pénsum un diplomado en Calidad de Software. Según el International Software Testing Qualifications Board (2020), República Dominicana ocupa menos del 3% de certificados en toda Hispanoamérica en su programa Certified Software Tester, es uno de los más prestigiosos en el área de calidad, dando a demostrar que todavía es mucho el trayecto por recorrer en esta área en el país.

Marco conceptual

Proceso: Según la International Organization for Standardization, ISO (2015), un proceso es un conjunto de actividades que están relacionadas entre sí y que transforman los elementos entrantes para obtener elementos de salida.

Software: Es un conjunto de programas de computación, procedimientos, reglas, documentación y datos que pertenecen a las operaciones de un sistema computacional (Institute of Electrical and Electronics Engineers, 1983).

Calidad: Grado en que las características inherentes a un objeto (producto, servicio, proceso, persona, organización, sistema o recurso) cumple con los requisitos (International Organization for Standardization, 2015).

Madurez de Proceso: Es un estado de óptimo desempeño, al cual un proceso puede llegar luego de pasar por etapas o niveles de madurez.

Factoría de Software: Es una empresa que se dedica al desarrollo de software como actividad comercial.

Defectos: Imperfección en un componente o sistema que puede causar fallos al desempeñar las funciones requeridas (International Software Testing Qualification Board, 2011).

Metodología: Es un conjunto de procedimientos que se utilizan con el fin de alcanzar un objetivo.

Producto de Software: Es un producto diseñado para un usuario (Institute of Electrical and Electronics Engineers, 1983).

Indicadores: Son comparaciones utilizadas para construir medidas cuantitativas u observaciones cualitativas.

Desarrollo Evolutivo: El desarrollo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndose a un refinamiento en las diferentes versiones (Sommerville, 2011).

MiPyMEs: Las micro, pequeñas y medianas empresas son agentes con cultura emprendedora.

Metodología

2.1. Tipo de investigación

El tipo de investigación será descriptiva debido a que se recolectarán las características y metodologías que comprende el proceso de calidad de software de Wepsys S.R.L, lo que permitirá que se haga un análisis en vista de lograr una propuesta que mejore el actual proceso.

Es una investigación documental puesto que se realizará un marco teórico, el cual servirá de apoyo a la investigación.

También se considera una investigación de campo ya que la información recaudada acerca del proceso se levantará aplicando métodos de investigación a los mismos colaboradores de Wepsys S.R.L.

2.2 Métodos de investigación

Los métodos que se utilizarán en esta investigación son:

2.2.1 Método Inductivo: El método inductivo es una estrategia de razonamiento que se utiliza para poder obtener conclusiones generales partiendo de hechos particulares. Es el método científico más usado.

El método inductivo es aquel procedimiento de investigación que pone en práctica el pensamiento o razonamiento inductivo. Este último se caracteriza por ser ampliativo, o sea, generalizador, ya que parte de premisas cuya verdad apoya la conclusión, pero no la garantiza.

El razonamiento inductivo consiste, así, en una forma de hipótesis que, a partir de una evidencia singular, sugiere la posibilidad de una conclusión universal. Esto suele expresarse en términos de probabilidades, tendencias o posibilidades, ya que no es posible afirmar nada de manera rotunda, ya que existe más información vital que la contenida en las premisas.

Esta forma de razonamiento es muy valiosa, dado que incorpora la creatividad y permite arriesgar conclusiones innovadoras que, si bien no pueden demostrarse, sí pueden someterse a consideraciones, pruebas y mecanismos de validación que, posteriormente,

conduzcan a la verdad. Por eso, el método inductivo forma parte del método científico, dado que sirve para expandir el conocimiento del mundo real que tenemos.

El origen moderno del método inductivo se remonta al siglo XVII y a la obra del filósofo inglés sir Francis Bacon (1561-1626), particularmente a su Novum organum scientiarum ("Nuevos instrumentos científicos") de 1620, donde precisó las reglas del método científico.

Se contrapuso a la tradición aristotélica del momento, en la que sólo se valoraban los razonamientos deductivos. Así, Bacon intentó demostrar la importancia de los razonamientos inductivos, pero aclarando que para llegar a una conclusión es necesario excluir muchas otras posibilidades.

Filósofos posteriores como David Hume (1711-1776), John Herschel (1792-1871) y John Stuart Mill (1806-1873) continuaron la tradición inaugurada por Bacon, y propusieron distintas formas de plantear la inducción con fines rigurosamente científicos.

Dicho esto, se utilizará este método para determinar el estado del proceso de calidad de software de Wepsys S.R.L.

2.2.2 Método Deductivo: El método deductivo es un proceso que parte de afirmaciones o principios generales para llegar a conclusiones particulares. Este es el método que se utilizará para plantear las mejoras al proceso de calidad partiendo del resultado obtenido en la investigación.

2.2.3 Método Cualitativo:

Los métodos de investigación cualitativos se utilizan con el objetivo de comprender el significado de un fenómeno. El rigor científico en estos métodos se basa en la credibilidad, la confiabilidad, la transferibilidad y la consistencia general. Los investigadores recolectan datos en el sitio donde se experimenta la problemática o la caso bajo estudio. La misma información será recolectada en el mismo sitio del caso de estudio, es decir, Wepsys S.R.L.

2.3 Técnicas de investigación

Se utilizarán las siguientes técnicas de recolección de datos:

2.3.1 Entrevista: La entrevista permitirá conocer cuál es la situación actual del proceso de Calidad de Software según los colaboradores, cuáles son las problemáticas a la hora de utilizar el proceso y determinar qué tanto estas afectan al negocio.

2.3.2 Encuesta: Permitirá hallar patrones y determinar cuáles son las oportunidades de mejora, además de mostrar la percepción de satisfacción del proceso actual.

2.3.3 Observación: Esta técnica se utilizará para adquirir información con relación a cómo funciona el proceso, desde un punto de vista objetivo con el fin de realizar un análisis.

Bibliografía

Awad, M. (2005). A Comparison between Agile and Traditional Software Development Methodologies. Perth, Australia.

Bauer. (1972).

Beck, K. (2000). Extreme Programming Explained. Boston, Massachusetts: Addison-Wesley Longman Publishing Co.,Inc.

Boehm. (1976). In Boehm, Software Engineering.

Boehm, B. (1988). A Spiral Model of Software Development and Enhancement.

Establecimientos, R. N. (2014). Estadísticas de Empresas TIC. Santo Domingo, República Dominicana.

Institute of Electrical and Electronics Engineers. (1983). IEEE 729.

International Organization for Standardization. (2015). ISO 9001.

International Software Testing Qualification Board. (2011). Syllabus - Certified Software Tester Foundation Level.

International Software Testing Qualifications Board. (2020). Distribución de certificaciones ISTQB en Hispanoamérica.

Kniberg, H., & Skarin, M. (2014). Kanban y Scrum: obteniendo lo mejor de ambos. C4Media Inc.

Laínez, J. R. (2014). Desarrollo de Software Ágil: Extreme Programming y Scrum. CreateSpace Independent Publishing Platform.

MiPymes, O. (2020). Conoce como las MiPymes del sector TIC prosperan en la nueva era digital. Santo Domingo, República Dominicana.

Pressman, R. (2002). Ingeniería del software: un enfoque práctico 5ta Ed. Madrid: McGraw-Hill.

RAE. (s.f.). Software.

Singh, Y. (2011). Software Testing. In Y. Singh, Software Testing.

Sommerville, I. (2011). Ingeniería de software (9na Edición ed.). Naucalpan de Juárez, Estado de México, México: Pearson Education, In.

Trigas, M. (2012). Metodología Scrum.

Webster's New Intercollegiate Dictionary. (1981).

Esquema Preliminar de Trabajo de Grado

Agradecimientos

Dedicatoria

Resumen ejecutivo

Introducción

Capítulo I.- Marco teórico referencial

1.1 La ingeniería de software

1.1.1 Concepto

1.1.2 Antecedentes

1.1.3 El ciclo de vida de software

1.1.4 Metodologías de desarrollo

1.1.5 Desarrollo y Calidad de Software en República Dominicana

1.2 Calidad de software

1.2.1 Concepto

1.2.2 Importancia

1.2.3 Proceso de aseguramiento de calidad de software

1.2.4 Las pruebas de software

1.2.5 Actividades del ciclo de pruebas

1.3 CMMI

1.3.1 Concepto

1.3.2 Antecedentes

1.3.3 Constelaciones de CMMI

1.3.4 Niveles de CMMI

1.3.5 Áreas de procesos de CMMI

1.4 Aseguramiento de la calidad de procesos y productos (PPQA)

1.4.1 Concepto

1.4.2 Beneficios

1.4.3 Objetivos

1.4.4 Forma de evaluación

Capítulo II.- Aspectos metodológicos

2.1. Tipo de investigación

2.2. Métodos de investigación

2.2.1. Método Inductivo

2.2.2. Método Deductivo

2.2.3. Método Cualitativo

2.3. Población

2.4. Muestra

2.4.1. Tamaño de la muestra

2.4.2. Tipo de muestra

2.5. Técnicas de investigación

2.5.1. Entrevista

2.5.2. Encuesta

2.5.3. Observación

Capítulo III.- Análisis detallado sobre el estado actual del proceso de calidad de software

Capítulo IV.- Propuesta de mejoras al proceso de calidad de software

Conclusión

Recomendación

Bibliografías

Anexos o apéndices