Universidad Autónoma del Estado de México

Centro Universitario UAEM Valle de Chalco

PRUEBAS PARA MEJORAR LA CALIDAD DEL SOFTWARE EN SOFTTEK

MEMORIA DE EXPERIENCIA LABORAL

QUE PARA OBTENER EL TÍTULO DE

LICENCIADA EN INFORMÁTICA ADMINISTRATIVA

P R E S E N T A

PATRICIA LOURDES BOLAÑOS PERAL

ASESOR:
DR. EN C. JOSE LUIS CASTILLO MENDOZA

Revisor:
DR. EN I. EN SIS. SAMUEL OLMOS PEÑA

Revisor:
M. EN C.C. LUIS MIGUEL DE LA CRUZ FLORES

VALLE DE CHALCO SOLIDARIDAD, MÉXICO AGOSTO 2021.

CUVCH

AGRADECIMIENTOS

Quiero agradecer a Dios por permitirme estar en una familia que me apoyo en todo momento y me alentó a culminar mis estudios

Agradezco a mis padres por ser parte fundamental y mi guía durante todo momento, porque gracias a su apoyo incondicional que me han brindado a su esfuerzo y aliento eh podido cumplir uno de mis más grandes anhelos.

Agradezco a mi asesor y revisores ya que con su experiencia, conocimiento, motivación, tiempo y dedicación me guiaron para poder culminar con el trabajo escrito de memoria de experiencia laboral.

Agradezco a la universidad Autónoma del Estado de México y a todos los docentes involucrados en mi formación académica ya que con su sabiduría paciencia y enseñanzas lograron mi crecimiento académico y profesional, y me motivaron para crecer día con día.

DEDICATORIA

A Dios por haberme ayudado durante estos años, a terminar mi carrera no fue nada fácil ya que el sacrifico fue grande, pero tú siempre me diste la fuerza necesaria para continuar y lograrlo, este triunfo te lo debo a ti ya que siempre estuviste junto a mí.

A mis padres por darme la vida y estar siempre presente en todo momento y alentarme cuando lo necesite, gracias por sus consejos y las fuerzas que me han dado durante lo largo de mi vida y por formarme porque gracias a ustedes soy la persona que soy hoy en día gracias por ser el motor de mi vida y por los sacrificios que han hecho por mí.

A mis hermanos por ser un ejemplo a seguir y darme fuerzas para lograr culminar la carrera, por alentarme y darme consejos que me han servido para ser mejor persona y alcanzar mis metas.

A mi asesor de tesis Dr. José Luis Castillo por el apoyo mostrado y la paciencia que me ha dado.

A mis revisores por las contribuciones aportadas en la realización de mi trabajo de titulación por su disposición apoyo y paciencia mostrada.

A mis amigas por estar siempre presente y apoyarme en todo momento por siempre estar junto a mí y aconsejarme y sobre todo por compartir sus conocimientos y por todas las experiencias inolvidables que hemos compartido.

A todos y cada uno de los profesores que formaron parte de mi formación profesional y por trasmitirme su conocimiento.

A la Universidad por brindarme la oportunidad de ser parte de ella, ahora puedo decir que soy orgullosamente UAEM.

PRUEBAS PARA MEJORAR LA CALIDAD DEL SOFTWARE EN SOFTTEK

ÍNDICE

I. RESUMEN

El presente trabajo es el resultado de la experiencia laboral adquirida durante 3 años de trabajo dentro de la empresa Softtek, S.A. de C.V (Sociedad Anónima de Capital Variable) - IT (Tecnologías de la información), *Services and Business Process Solutions*, quien es un proveedor global de servicios orientado a procesos de TI, Softtek mejora el tiempo de entrega de soluciones de negocio, reduce costos en las aplicaciones existentes, entrega aplicaciones mejor diseñadas y probadas, y produce resultados predecibles para grandes empresas en más de 20 países, en la que he laborado en el área de pruebas, la cual es fundamental dentro del proceso de desarrollo de software.

Las pruebas representan el último bastión desde donde puede valorarse la calidad y, de manera más pragmática, descubrirse errores. Pero las pruebas no deben verse como una red de seguridad. Como se dice: “no se puede probar la calidad, si no está ahí antes de comenzar las pruebas, no estará cuando termine de probar”. La calidad se incorpora en el software a lo largo de todo el proceso de ingeniería del software. La adecuada aplicación de métodos y herramientas, revisiones técnicas efectivas, y gestión y medición sólidas conducen a la calidad que se confirma durante las pruebas (Roger, 2010, p.385).

La empresa atiende diversas cuentas con giros muy distintos telefonía, seguridad social, aseguradoras, tiendas departamentales, bancos etc., en donde me han asignado en alguna de ellas., y a su vez en distintos proyectos y de acuerdo a las vivencias tenidas en cada proyecto se ha logrado detectar algunas problemáticas como por ejemplo el retraso en los tiempos de entrega o los re trabajos ocasionado a que muchas veces el usuario no conoce por completo su negocio, por lo que piden cambios constante mente y los analistas no bajan la información a todo el equipo de desarrollo, lo que se

han logrado mitigar derivada de acciones implementadas que se describirán más adelante además, que se abordaran las principales habilidades que se deben de tener para lograr desempeñarse en el puestos y las principales actividades desarrolladas en el área, se describirán el proceso de prueba, las técnicas y herramientas implementadas. Cabe mencionar que parte de las competencias adquiridas se obtuvieron con esfuerzo, dedicación y preparación dentro de la Centro Universitario UAEM Valle de Chalco, logrando así aplicar los conocimientos obtenidos en el ambiente laboral, y ejercer en el área de pruebas lo que ha permitido aprender nuevos conocimientos y desarrollar experiencia dentro del área.

II. IMPORTANCIA DE LA TEMÁTICA

Las pruebas de software (Testing), tienen como objetivo descubrir los errores que se cometieron de manera inadvertida conforme se diseñó y construyo el software, son un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática por lo tanto se debe definir una plantilla para las pruebas de software que son un conjunto de pasos en que se deben de incluir técnicas y métodos específicos del diseño de casos de pruebas (Roger, 2010, p.384).

Las organizaciones dependen del software diariamente, derivado de los sistemas de información que son una herramienta de trabajo dentro de ellas, por ejemplo: telefonía, seguridad social, programas bancarios, etc. Además, que forman parte de nuestra vida cotidiana.

Es importante hacer mención que los productos de software, sistemas y aplicaciones, son desarrolladas e implementadas por personas especializadas en el área, sin embargo, en cualquiera de las etapas de desarrollo se puede inyectar algún *bug* (error), ya sea por distracción, mal entendimiento del requerimiento, documento de requerimientos con huecos funcionales, distracción al codificar etc., incluso los errores pueden permanecer sin ser descubierto lo que ocasiona un sistema con defectos, ay que recordar que un desarrollador codifica de acuerdo a la documentación, y esta a su vez es creada por los analistas entre otras (Mera, 2015).

Es por lo anterior que es importante realizar pruebas en los productos terminados de desarrollo de software, ya que no ejecutarlas o las malas prácticas en la ejecución de pruebas pueden llegar a producir fallos en los sistemas cuando son puestos en producción y ocasionar daños como pérdidas económicas, mala reputación, desconfianza ante el cliente e inclusive llegar a provocar pérdidas humanas, eso depende del tipo de software/Aplicación/Sistema que es probado.

Es importante hacer mención que los costos de eliminar defectos se incrementan conforme pasa el tiempo durante el cual el defecto permanece en el sistema, la detección de bugs en etapas tempranas permite la corrección de estos a costos reducidos, no es lo mismo encontrar un defecto durante el análisis a encontrarlo en producción (Campos, 2015, pag.7).

Las pruebas (testing) aseguran el correcto funcionamiento de la aplicación de acuerdo con los requisitos solicitados por el cliente además permite prevenir fallos en producción y ganar la confianza del cliente.

Uno de los 7 principios del ISTQB (Comité Internacional de Calificación en Pruebas de Software), indica que “la pruebas son una manera eficaz de encontrar errores, pero no puede probar que no hay defectos” (ISTQB, 2018, pag.24), y son realizadas para alcanzar la calidad del *Software* que espera el cliente. La IEEE (Instituto de Ingenieros Eléctricos y Electrónicos), define calidad de software como "el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario", denotando que el énfasis radica en los requisitos específicos del sistema y en la búsqueda de la satisfacción del cliente (Callejas y Álvarez, 2017, pag.237).

Sin embargo, una de las problemáticas presentadas comúnmente es que el usuario no conoce por completo el negocio, inclusive en el trascurso del análisis o ejecución de pruebas se dan cuenta que la funcionalidad está incompleta por lo que, constante mente solicitan cambios al área sobre la documentación cerrada lo que retrasa los tiempos de entrega, esto deriva que se realicen controles de cambio que muchas veces no es notificada a todo el equipo involucrado lo que, provoca confusión en las diferentes áreas.

Es importante que el equipo tenga claro las condiciones que busca el usuario en el software, además, es importante realizar comparaciones sobre proyectos anteriores con la finalidad de tener referentes en la estructura, la organización, los tiempos y la calidad, además, permite que sobre la experiencia

vivida se puedan tomar decisiones. Es importante que el gerente del proyecto esté centrado en lo que los clientes buscan en el software y que satisfagan sus necesidades, y la calidad es parte importante; por eso se debe seguir una metodología la cual incluye principios que, a través de un plan de pruebas, sean guía para el equipo y que este realice una optimización de las herramientas, técnicas y conocimientos. Además, se debe incluir al cliente como elemento valioso de todo proyecto en el proceso de pruebas de calidad, para que se entere de los avances, conozca los posibles fallos y lo que no se considera fallo; así mismo, pueda reportarlos si se presentan en producción (Mera, 2016, p. 173).

Por eso es importante dar a conocer la información que involucra el proceso de pruebas para mejorar y controlar la calidad de software, y con ello que dentro del ciclo de desarrollo se conozcan los roles y responsabilidades de forma correcta. Además, es primordial recordar que los productos de desarrollo de software son elaborados por personas y por lo tanto se puede tener equivocaciones. Así que el testing, es un factor determinante para lograr el éxito del proyecto, por lo que es importante trabajar con una metodología y técnicas y herramientas que nos orienten para la entrega de un “software con la mejor calidad posible, esperada por el cliente de acuerdo con su solicitud”.

Se aborda el conocimiento adquirido durante más de tres años de trabajo en Softtek, se explica el papel de las pruebas dentro del proceso de desarrollo de Software, hablaremos de la metodología V utilizada dentro de la empresa asignada, las técnicas, y los distintos tipos de pruebas ejecutadas dentro del proceso para lograr la mayor calidad posible del software, se abordaran algunas problemáticas y soluciones, durante el trabajo se abordara el tema de la calidad del software y el ¿Por qué es importante que un proyecto de software sea aprobado por el área de pruebas?.

III. DESCRIPCIÓN DEL PUESTO O EMPLEO

Se dará a conocer los antecedentes y principales actividades en las que funge la empresa, así como su principal estructura organizacional con la que trabajo en la cuenta asignada, se explicaran los roles y responsabilidades de cada área involucrada dentro del desarrollo de software, dando énfasis al puesto de pruebas.

Antecedentes de la empresa

En diciembre de 1982, *Softtek* fue fundada en México como una pequeña compañía de servicios de TI (Tecnologías de la información). En 1997 Softtek introdujo el modelo de servicios *Nearshore* (forma de subcontratación que se refiere a los servicios desde una ubicación adyacente o cercana), con la creación del Centro Global de Entrega en Monterrey y México, el primero en su tipo en toda Latinoamérica. Aunque la compañía ha crecido enormemente desde su origen, promovida por una cultura corporativa única, esta tendencia se aceleró luego de que la actual presidente y CEO, tomara posición como CEO en el año 2000. En el 2003, adquirió el Centro Global de Desarrollo, en México y expandió su portafolio de aplicaciones y servicios al combinar las capacidades de los dos jugadores más fuertes de *Nearshore* en México. Se transformó en un proveedor global de soluciones de TI (Tecnologías de la Información), y procesos de negocio con 12,000 colaboradores a través de 30 oficinas en Norteamérica, Latinoamérica, Europa y Asia (Softtek, 2019).

Principales servicios de Softtek

La compañía en la que laboro se dedica a las tecnologías de la información, y tiene un amplio portafolio de productos y soluciones que transforman cualquier negocio, ayudando a empresas a evolucionar de manera fluida y constante desde la ideación y construcción hasta la ejecución de estrategias digitales. A continuación, se describen algunos de los servicios brindados, en donde como ingeniero de pruebas aporto valor en el área:

Modernización de aplicaciones

En lugar de deshacernos de activos valiosos y ralentizar la transformación, reutilizamos las aplicaciones heredadas utilizando patrones y plataformas arquitectónicas modernas, mitigamos los riesgos de obsolescencia de la tecnología, mejoramos la usabilidad, la seguridad y el mantenimiento. Nuestro enfoque aprovecha lo mejor de sus activos y hace que el funcionamiento sea impecable como componente de una base digital moderna. Es una actividad en la cual como área de pruebas nos involucramos, ya que es importante entregar un software que cumpla con la expectativa del cliente y al ejecutar pruebas evaluamos el comportamiento del software que ha sido reutilizado.

Desarrollo de *software*

Proveedor más grande de servicios de desarrollo donde, se crean aplicaciones de próxima generación utilizando *Agile* (enfoque centrado en el cliente para definir, construir y liberar un flujo continuo de productos y servicios valiosos para clientes y usuarios), y *DevOps* (enfoque cada vez más común para la entrega de software, por el cual los equipos de desarrollo y operaciones colaboran para crear, probar, implementar y monitorear aplicaciones con velocidad, calidad y control). (IBM, 2017).

Maximizando la entrega, el compromiso del equipo y los objetivos de productividad. Si bien este es un servicio que ofrece Softtek, y que pueden contratar las empresas, así mismo eh trabajado con los equipos de desarrollo de la empresa, actualmente en la cuenta asignada el desarrollo de software es codificado por su personal interno, utilizando la metodología V y Scrum, esta última actualmente se ha implementado, por lo que es importante que como equipo de Testing (Pruebas), trabajamos con ellas y en conjunto con el equipo para entregar un producto con calidad, que deje satisfecho a los clientes.

Pruebas de *Software* y Calidad (QA)

Es una práctica sólida y completa donde aseguramos la calidad y ejecutamos pruebas que logra identificar con exactitud e integridad el nivel de los productos de software. Ayudando a las organizaciones a mejorar la productividad y acelerar el tiempo de comercialización y la liberación de los productos de software, que cumplan con las expectativas del cliente en términos de calidad y experiencia de los usuarios, al mismo tiempo que se obtiene un importante ahorro en los costos (Softtek, 2019).

Estos son tan solo algunos de los servicios que Softtek proporciona a las organizaciones y en donde como pruebas aporto valor para brindar un servicio de calidad para que los clientes se encuentren satisfecho y tengan una buena imagen de la compañía.

Organigrama general

A continuación, se presenta la estructura principal de la empresa de la alta gerencia en donde cada uno de ellos forma parte fundamental para que Softtek se encuentre posicionado como una de las mejores empresas de tecnología, haciendo énfasis a la dirección de operaciones que es bajo la dirección en donde se encuentra mi área “Testing” (Pruebas de Software).

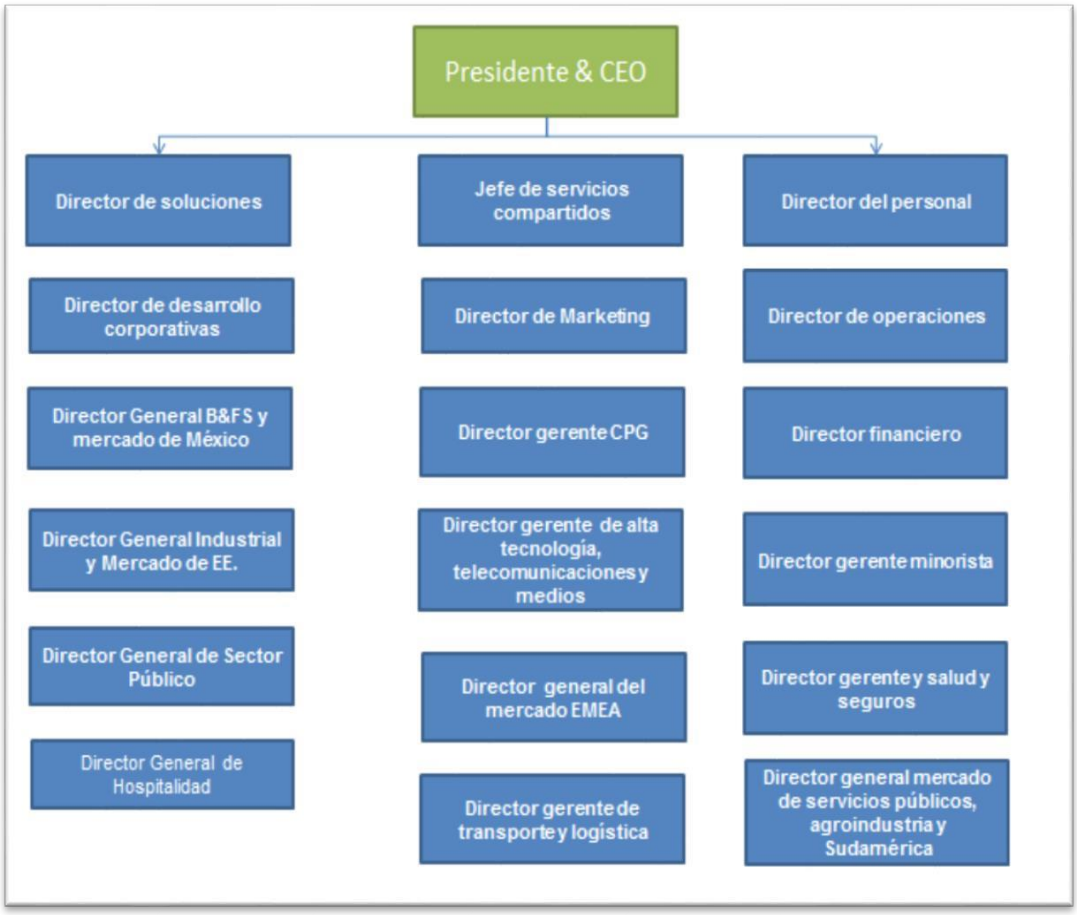

Figura 1. Organigrama general, muestra el equipo de la alta gerencia (Softtek, 2019).

Es importante hacer mención del equipo se encuentra involucrado en los proyecto de desarrollo de software, el cual abarca diferentes áreas una de ellas es testing, donde garantizamos, controlamos y mejoramos la calidad de la aplicación, y donde generamos confianza hacia los clientes, sin embargo, para lograr el objetivo es importante trabajar en equipo y tener comunicación con todos los miembros del equipo para la toma de decisiones, a continuación se muestra el organigrama que involucra a todo el equipo de trabajo con el que me relaciono y en donde cada rol aporta valor para lograr entregar un producto con calidad:

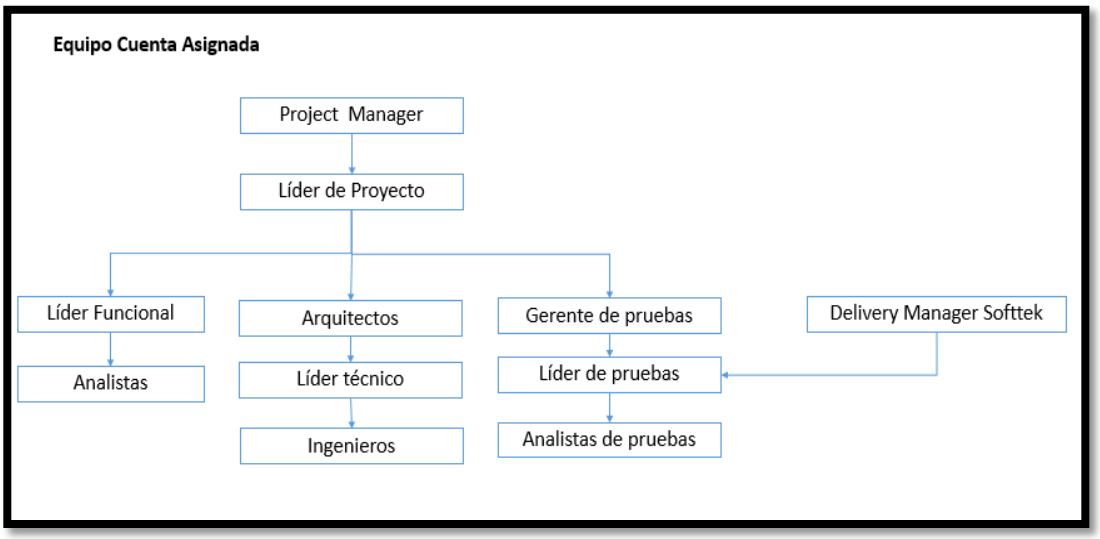

Figura 2. Organigrama, que muestra la estructura del equipo de trabajo (Creación propia, 2019).

Delivery Manager o Gerente de entregas es quien se encarga de que la empresa sea efectiva y su principal enfoque es que se terminen los proyectos en los que estamos trabajando, por ello puede pedir priorizar tareas (Study, 2020), además busca capacitarnos constantemente sobre herramientas, que la cuenta requiera implementar o sobre cursos que aporten valor a mi rol, entre sus funciones principales se encuentran:

- Buscar e incorpora nuevas cuentas lo cual es importante ya que de esto derivan los proyectos de desarrollo de software.
- Da apoya a líderes técnicos a medida que innovan nuevas formas de entregar productos de software a los clientes.
- Contrata a personal capacitado para el proyecto y a su vez es quien puede considerarnos para futuras cuentas y proyectos.
- Crea buena relación con el cliente.
- Trabaja en conjunto con los gerentes de producto para definir el plan del proyecto de desarrollo con el cual estaremos trabajando.

Este rol es muy importante, ya que es quien se encarga de traer más cuentas y quien puede considerarnos para futuros proyectos, además, es mi contacto directo con la empresa y a quien reporto las actividades y proyectos que tengo asignados, además, da seguimiento a las actividades que debo realizar en la empresa como registro de horas de las tareas realizadas durante la semana etc.

Project Manager o Gerente de Proyectos es quien lidera al proyecto, para lograr los resultados esperados, debe tener claro el propósito, además, es quien se encarga de dar solución a cada problema que pueda presentarse en los cambios del plan inicial (Study, 2020), entre sus responsabilidades que tiene son:

- Desarrolla el contrato entre el equipo de proyecto y cada parte interesada incluyendo los objetivos, los costos y la fecha límite de entrega.
- Proporciona elementos visuales ejemplo diagramas que describan los objetivos de cada proyecto.
- Asignar recursos humanos, materias primas a las diferentes áreas del proyecto.
- Pide retroalimentación y comunicar los cambios y problemas que puedan llegar a surgir.

Es importante tener comunicación con el PM, desde que inicia el proyecto y mantenerlo informado de todo lo que suceda durante todas las etapas del proyecto ya que, debe dar solución, o bien, presionar a las áreas responsables para no retrasar los tiempos de entrega.

Líder de proyecto debe de estar al tanto de lo que se ha logrado y lo que no, debe delegar todo el trabajo y abstenerse de realizar trabajo técnico salvo que sea para capacitación o sirva de entrenamiento practico para alguno de nosotros, mide el tiempo que falta para la culminación del proyecto y si existiera algún problema puede realizar nuevamente el cálculo de tiempo necesario para finalizar con el proyecto, debe de estar alerta para reconocer alertas de peligro (Llorens, 2005, p.108-111) etc. Además, es quien influye en cada uno nosotros como miembros del equipo para lograr los objetivos del proyecto, reconoce e inicia cambios en el proyecto para llevarlo por el buen camino, nos alienta y motivo a que demos solución de los problemas que puedan surgir, fomenta la colaboración etc.

Líder funcional es responsable de coordinar en conjunto con el usuario la definición de las necesidades requeridas y generar los requerimientos. Funge como cliente final ante el equipo técnico. Define las funcionalidades que debe contener la solución, de forma que cumpla y satisfagan las necesidades solicitadas por el cliente, es importante tomar en cuenta que este define la solución y no el cómo debe realizarse, este rol es quien tiene todo el contexto del negocio y quien baja la información a su equipo de trabajo para que la analicen y se generen los requerimientos los cuales son importantes para mi área ya que son mi base de trabajo, él tiene a su cargo a los analistas de negocio a continuación se describen (ACM, 2020).

Analistas de negocio como su nombre lo indica el rol de analista será responsable de analizar los procesos de negocios, especificar los requisitos y diseñar la interfaz usuario. Debe estar muy relacionado con los interesados en el negocio para determinar claramente las funcionalidades del sistema (López y André, 2006, p.37), este rol es fundamental para nuestra área ya que en base a lo que el escribe en los documentos es con lo que trabajamos durante todo el proceso de pruebas, además, si tenemos dudas respecto a la funcionalidad plasmada debemos de acercarnos a clarificarlas.

Arquitecto es responsable de la definición y diseño de la arquitectura. Entre sus competencias está: Definir la arquitectura de los sistemas tomando las decisiones de diseño de alto nivel y estableciendo los estándares técnicos, incluyendo plataformas, herramientas y estándares de programación, teniendo en cuenta los requisitos funcionales, no funcionales y las necesidades del negocio con el cual tenemos constante comunicación ya que es el que se encarga de toda la definición, diagramas y quien nos ayuda a tener contexto de como los sistemas se relacionan, nos aclaran dudas en caso de que sea necesario (Cessi, 2020).

Líder Técnico responsable de alumbrar y dar forma a la arquitectura planteada por el Arquitecto de software, haciendo énfasis en las especificaciones que la arquitectura no presenta en detalle, es básicamente responsable de un proceso de refinamiento, en el cual se toma la visión del arquitecto de software y se especifica en detalles más prácticos para el equipo de desarrollo, y en el caso del surgimiento de retos y/o problemas, es el responsable de definir las metodologías y las técnicas para abordarlos. Como última parte, y la más importante, un líder técnico tiene una meta diaria que nunca puede olvidar, y es la de brindar ayuda técnica continua y proactiva a todos los desarrolladores del equipo, con el fin de garantizar el éxito de cada uno de ellos en el proyecto, aunque como tester no tenemos mucha comunicación con él, es parte fundamental cuando se tienen defectos crítico y su equipo no da una solución, es la persona clave para escalar todos estos temas y da apoyo para que el proceso

sea más ágil (Tovar, 2018).

Ingeniero (Desarrollador) es quien se encarga de escribir el código de acuerdo con las especificaciones proporcionadas por los líderes de desarrollo que deben de estar basados en los requerimientos funcionales proporcionados por los analistas y es parte fundamental cuando se inicia con la ejecución de pruebas e inclusive en atapas anteriores ya que es la base para que los objetivos sean cumplidos (Bogue, 2005).

Él es el responsable de informarnos para que iniciemos con la ejecución de pruebas apegándose a las fechas estipuladas en el plan de pruebas que fue definido, aunque puede dar el caso de que no se cumpla con ello, por lo que deben de informarlo para que el equipo ajuste fechas y se lleguen a los acuerdos necesarios.

Coordinador de pruebas coordina y dirige el desarrollo e implementación de estrategias para la planeación y ejecución de pruebas del equipo de trabajo, con el fin de garantizar la calidad del software, aplicando el uso de buenas prácticas, para satisfacer las necesidades de la organización, es quien en conjunto con el líder asigna los proyectos de pruebas a los miembros del equipo y apoya a la solución de problemas dentro del área.

Líder de pruebas gestionar y administrar los proyectos, tiene a su cargo una cartera de proyectos asignados y es su responsabilidad supervisar el servicio de pruebas gestionando analistas externos y siendo responsable del cumplimiento de plazos y calidad de pruebas realizadas. En conjunto se coordinan las pruebas técnicas y ambientaciones en ambientes no productivos, además, es quien nos asigna los requerimientos o casos de uso en conjunto con el coordinador, los cuales serán mi base de trabajo, es decir, que realizare el análisis, diseño, ambientación, ejecución etc., y es la persona con la que tengo mayor comunicación, ya que es a quien le reporto todos los temas que van surgiendo y nos apoya a presionar, canalizar e investigar en caso de que

sea requerido. (Reqlut, 2018).

Como Analista de pruebas (tester) me encuentro dentro del área de testing, en donde participo en cada una de las etapas del desarrollo de software desde análisis de requerimientos, diseño, desarrollo, pruebas, operación y mantenimiento, y mi función principal es mejorar y controlar la calidad del *software,* ejecutando pruebas estáticas, es decir, se realizan sin la ejecución del software y se llevan a cabo al revisar la documentación proporcionada por los líderes y la finalidad es encontrar inconsistencias que deben de ser revisadas y corregidas por los analistas, así como pruebas dinámicas que son realizadas al ejecutar el software con la finalidad de validar el funcionamiento correcto de acuerdo a la documentación con el objetivo de generar confianza hacia el cliente, entregando un sistema que cumpla con lo solicitado.

Es importante ejecutar pruebas de software, debido a que los documentos utilizados para la codificación y diseño de pruebas, son realizados por seres humanos por lo que, pueden tener equivocaciones, inclusive en la etapa de desarrollo al generar el código se puede inyectar defectos, de ahí la importancia de ejecutar pruebas que permitan identificarlos y sean reportados hasta su solución, ya que el saltar esta etapa dentro del proceso de desarrollo de software, puede llegar a ocasionar que al poner en producción el software, presente fallos y que no funcione de acuerdo a lo especificado por el cliente esto podría llegar a ocasionar mala reputación de la empresa, desconfianza por parte de los clientes e inclusive pérdidas económicas, recordemos que los costos se incrementan puesto que, no es lo mismo solucionar un defecto en la etapa de análisis que en la de producción ya que, en esta ultima el software ya se encuentra en funcionamiento.

Y precisamente esta es una de las actividades más importante de mi perfil puesto que, me encargo de analizar, diseñar, ambientar datos, ejecutar e informar el resultado de pruebas del proyecto de software, que está a mi cargo a todo el equipo involucrado además que en metodologías tradicionales somos

intermediarios entre lo que construye desarrollo con respecto a lo que solicito el cliente, ya que a veces desarrollo tiene un contexto distinto y la solución no es la que esperaba el cliente, por lo que nuestro objetivo es asegurar que se eliminen los huecos entre ambos puntos de vista y con esto entregar un producto con calidad, asegurándonos que se cumple con lo que espera el cliente. A continuación, describo las funciones principales que desempeño como *tester*, las cuáles son importantes para proporcionar confianza y calidad en mi trabajo:

- Cuando tengo un documento de requerimientos o caso de uso asignado por el líder, inicio con la inspección del documento, esto quiere decir que inicio con la lectura del documento, con la finalidad de identificar huecos funcionales o inconsistencias dentro de ellos, los cuales debo de reportar y dar seguimiento hasta que se dé solución.

- Identifico la cobertura de pruebas o estimación de pruebas usando técnicas que me permitan obtener todos los posibles casos de prueba que comprobaran que el sistema cumpla con las especificaciones del cliente y la calidad esperada por ellos.

- Genero la matriz de pruebas funcionales la cual sirve como guía cuando se ejecutan los casos de prueba ya que, debe de contener todos los pasos y resultados esperados por cada caso de prueba además debe ser clara para que cualquier *tester* que ejecute, pueda realizarlo sin la necesidad de leer los documentos.

- Ejecuto todo el set de pruebas, y valido que la funcionalidad por cada caso de pruebas sea la esperada respecto a los requerimientos solicitados por el usuario, esta etapa considero es sumamente importante ya que se da a conocer si el producto funciona como se espera, y reporto diariamente los resultados obtenidos durante todo el periodo de ejecución.

- Tengo la capacidad de detectar *Issu* (defectos), y si los encuentros los levanto en las herramientas de trabajo adjuntando evidencia y doy seguimiento hasta su solución.

- Opero los sistemas o programas bajo prueba con la finalidad de conocerlo, explorarlo y probarlo con el objetivo de hacer todas las combinaciones necesarias para encontrar errores, tanto de funcionalidad como de negocio.

- Tengo claro que lo que se entrega al usuario final corresponde a lo que fue solicitado por el cliente por medio de los requerimientos, genero confianza de que el producto probado cumple con la funcionalidad especificada y hace lo que debe de hacer según lo descrito.

- Evalúo las capacidades de *software* y determino si cumple con la funcionalidad solicitada y si los resultados son los esperados.

- Doy panorama referente a la calidad del software y minimizo el nivel de desconfianza que pueda tener el usuario referente a los resultados de pruebas obtenidos.

Mi rol es importante dentro del desarrollo de software, ya que somos el área que evalúa que el producto se creó en base a lo solicitado por el cliente y que cumple con la funcionalidad requerida, para lograrlo debemos tener conocimiento del negocio y no tener miedo a explorar la aplicación para encontrar todos los errores en ella, a continuación, se muestran las cualidades con las que debo de contar para desempeñar el rol:

- Trabajo en equipo
- Trabajo en equipo
- Trabajo bajo presión.
- Meticuloso.
- Capaz de detectar errores.
- Crítico.
- Capacidad de adaptarse a cambios.
- Pasión por el producto.
- Desconfiado.
- Curioso.
- Atento.
- Detallista.
- Ético.
- Capacidad de comunicación.

Todas son de importancia, para ser un buen candidato para desempeñarme en el rol y sumar valor para lograr los objetivos del proyecto, sin embargo, una de las aptitudes que no puede faltar es el trabajo en equipo.

IV. PROBLEMÁTICA IDENTIFICADA

Es una empresa dedicada a brindar servicios de TI (Tecnologías de la información), en la que día a día atiende diversas cuentas. Se ha tenido participación en el área de calidad en varias de ellas, en donde estas se han dividido en diferentes proyectos. De acuerdo con las vivencias en cada uno de ellos se ha logrado detectar algunas problemáticas las cuales se describen:

Lo fundamental en la etapa de pruebas es validar la calidad del software, así como asegurar que se cumplan con el total de requerimientos especificados por el cliente. Buscando la confianza en el producto, ya que la aceptación del cliente se da cuando hay calidad en el producto final y ellos la miden por la cantidad de defectos encontrados y si no existiera podría hacer que el usuario tenga inseguridad, insatisfacción y que genere desconfianza. El usuario, es una pieza clave, aunque también puede llegar a ocasionar retrasos esto debido a que una de las problemáticas presentadas a menudo es que el usuario no conoce por completo el negocio, por lo que en el trascurso del análisis o ejecución de pruebas se dan cuenta que la funcionalidad está incompleta. Por lo que a menudo solicitan cambios al área de análisis es decir, afectación a la documentación cerrada (casos de uso, reglas de negocio o requerimientos etc.), esta problemática retrasa los tiempos de entrega del producto final debido a que el equipo de análisis debe de modificar la documentación afectada, lo que a su vez retrasa al equipo de testing, que debe de trabajar en los artefactos que ya habían sido cerrados es decir modificación de casos de prueba, matriz de diseño de casos de prueba etc, y el equipo de desarrollo debe de trabajar en el código de los cambios solicitados, esto ocasiona que la duración del proyecto sea extendido y no salga en las fechas comprometidas por la empresa, también es importante considerar que entre mayor sea la etapa en la que se dan cuenta de que no solicitaron cierta funcionalidad mayor es el costo, ya que no es lo mismo encontrar un defecto durante el análisis de documentación a encontrarlo cuando las pruebas están por finalizar, además entre mayor sea el tiempo de retraso duración del proyecto el consto aumentara.

Si bien los usuarios entienden los procesos de negocio ya que operan con ellos, incluso tienen contexto de los flujos principales del sistema y el funcionamiento que se desean implementar, no suelen tener en cuenta todos los flujos alternativos y las condiciones de error, así como los diferentes grupos de requerimientos que se relacionan entre sí por lo cual deben de trabajar de la mano con el equipo de desarrollo, además se debe de considerar que en ocasiones los usuarios que solicitan la funcionalidad no son los mismos que les dan seguimiento.

Una segunda problemática deriva de los controles de cambio que solicita el cliente, en donde muchas veces los analistas no distribuyen la información a todo el equipo involucrado (Desarrolladores, *Project Manager*, *Testers*, entre otros), lo que ocasiona confusión en las diferentes áreas, desarrollo incorrecto, pruebas incorrectas o incompletas y sobre todo retrabajos. Por ejemplo: cuando un analista hace modificaciones a los casos de uso o cualquier artefacto, es importante sea notificado todo el equipo, dado que el desarrollador codifica de acuerdo con la documentación y el área de pruebas se analiza, diseña y ejecuta con las mismas bases. Si el analista no distribuye la información en tiempo, ocasiona defectos y retrabajos ya sea a nivel código o en artefactos del área de pruebas (diseño de los casos de prueba, cobertura de pruebas y/o matriz de pruebas) e inclusive de la ejecución. Ya que en el momento en el que se detecta que los cambios son incorrectos, se debe volver a codificar y realizar las modificaciones correspondientes tanto en área de desarrollo como en el área de pruebas. Lo que ocasiona que no se inicien las pruebas en el tiempo estimado y/o que se retrasen los tiempos de entrega y estos se alarguen demasiado provocando que la calidad no sea aceptable o incluso se tiene que realizar sobre esfuerzo para hacer la entrega en el tiempo estimado por el equipo.

V. INFORME DETALLADO DE LAS ACTIVIDADES

Las pruebas de *Software* es un proceso que ayuda a identificar posibles fallos de implementación, usabilidad y calidad etc., de un sistema por lo cual son importantes para mejorar y controlar la calidad del *Software*, para ello se debe efectuar pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones solicitadas por el cliente es importante tener presente que todo proyecto tiene como objetivo producir software con la mejor calidad posible, que cumpla, y si puede supere las expectativas del usuario (Fossati, 2016).

Uno de los objetivos principales como ingeniero de pruebas es encontrar defectos, aunque pueda pensarse todo lo contrario, sin embargo, los sistemas son tan complejos que tal demostración resulta imposible en la práctica, además de que los sistemas son desarrollados por humanos quienes pueden llegar a cometer errores y para poder demostrar que el software no está libre de ellos, es importante probarlo con todos los datos de entrada y las salida esperadas o mediante especificaciones formales además, debemos de tener claro el alcance de las pruebas y tener contexto del negocio.

Es importante apegarse a una metodología que funja como marco de trabajo, y que sirva de guía para cumplir con el objetivo un software con la mejor calidad posible, a continuación, se explicará la metodología V que significa Validación & Verificación la cual es mi guía para la realización de pruebas.

Pruebas a través del modelo en V

Un modelo de ciclo de vida de desarrollo de *software* describe las actividades realizadas en cada etapa de un proyecto de desarrollo de software y cómo las actividades se relacionan entre sí de manera lógica y cronológica.

Existen diversos modelos de desarrollo de *software*, cada uno de los cuales tiene diferentes enfoques para la prueba. Sin embargo, la metodología V que significa verificación y validación es la utilizada en la cuenta asignada y es nuestra guía por lo que, debemos de tenerla presente y estar familiarizado con el ciclo de vida para la realización de nuestras actividades.

Probar es parte fundamental de la verificación y validación, por lo que necesitamos realizar actividades como, por ejemplo, revisión de la documentación etc., para controlar la calidad del software. El modelo V integra el proceso de prueba durante todo el proceso de desarrollo, implementando el principio de pruebas tempranas. Además, el modelo incluye niveles de prueba asociados con cada fase de desarrollo correspondiente, que respalda aún más las pruebas tempranas (ISTQB, 2018, p.39). A continuación, se muestra la representación gráfica el modelo de ciclo de vida en V.

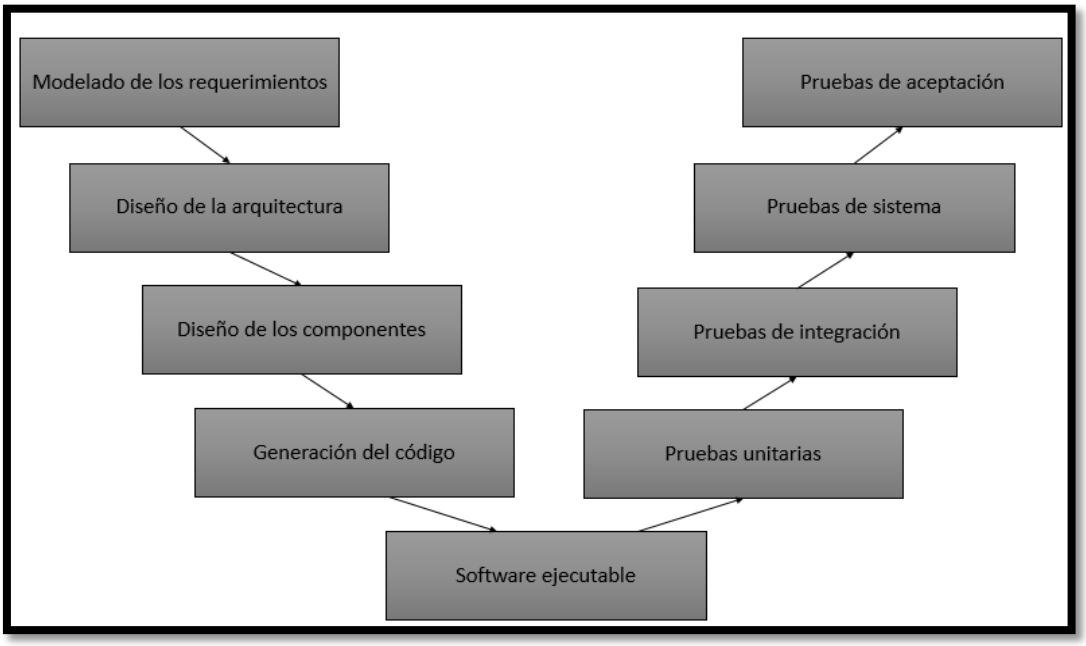

Figura 3. Modelo de ciclo de vida en V, (Roger, 2010).

En el diagrama muestra de lado izquierdo la verificación de cada una de las ramas, a continuación, se da una breve descripción de cada una de las fases:

- Modelado de los requerimientos: se realiza toda la documentación con las especificaciones del cliente, es una de las funciones principales del analista que son quienes se involucran con ellos para entender el negocio y es la base para que analice, diseñe y ejecute las pruebas.

- Diseño de la arquitectura: Es prácticamente definir la arquitectura y las interfaces que formaran parte de él, para mi área es importante ya que cuando revisamos la documentación, es decir, diagramas, bases de datos etc., de cómo va a interactuar y comunicarse el aplicativo nos da un mayor panorama de entendimiento.

- Diseño del componente: Se especifica la estructura de los componentes que formaran parte del programa.

- Generación del código: Se trata de escribir el código que será ejecutable.

- Software ejecutable: Es el programa listo para realizar las funciones de acuerdo con los requerimientos solicitados y es el momento en que entramos como área de pruebas para validar que él software cumpla con lo requerido por el cliente.

Cada nivel de desarrollo se verifica respecto al contenido del nivel que le precede para ello es importante definir este concepto, según ISO 9000 es la confirmación, mediante la aportación de evidencia objetiva de que se han cumplido los requisitos especificados (ISO, 2015), así bien de lado izquierdo podemos observar que en la fase diseño de la arquitectura se debe verificar que los requisitos de la primera fase han sido implementados correctamente y así sucesivamente con cada una de las fase. Como se muestra en la siguiente imagen:

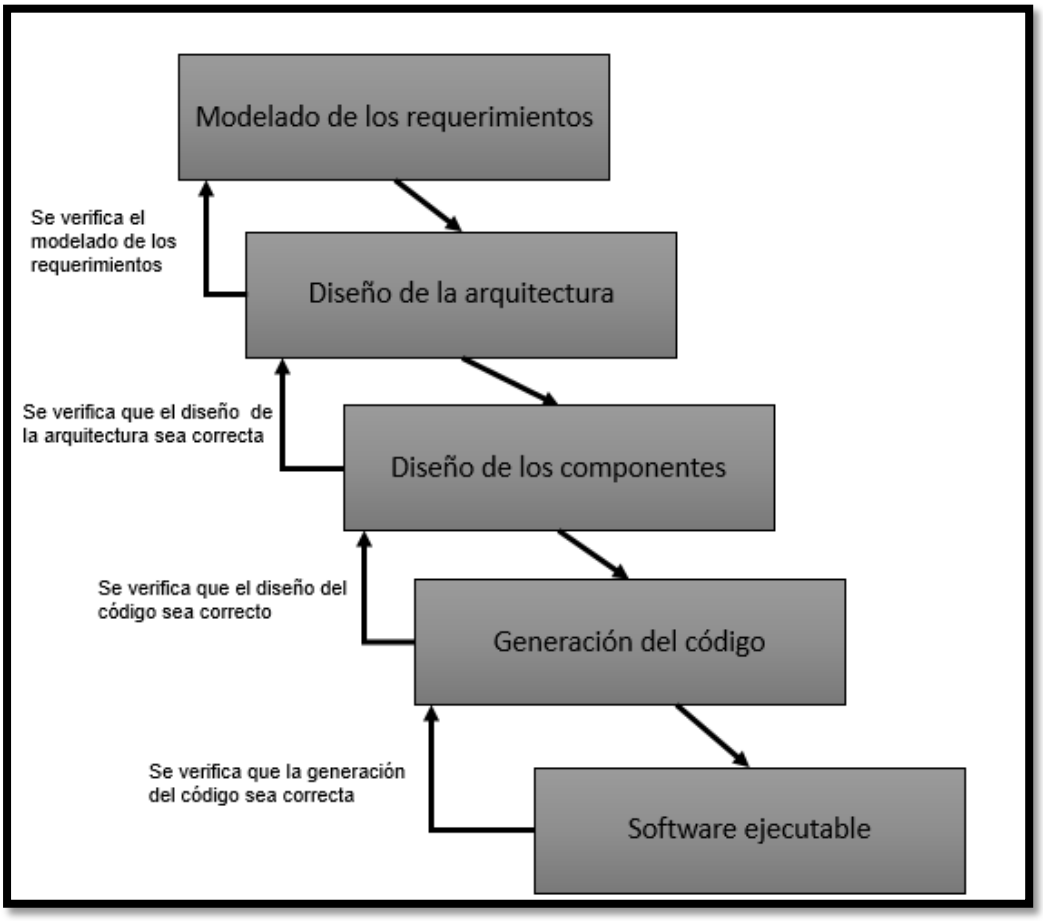

Figura 4. Verificación en el Modelo de ciclo de vida en V (Roger, 2010).

Mientras de lado derecho del diagrama se muestran las pruebas realizadas de acuerdo cada una de las fases, si bien no todas son realizadas por nuestra área, es parte fundamental para que cuando ejecutemos nuestras pruebas los issus encontrado sean mínimos. A continuación, se dará una breve descripción por cada una de las fases:

- Pruebas de componentes (unitarias): Se centran en los componentes que se pueden probar por separados y sus objetivos son encontrar defectos en ellos, verificar que los componentes son los diseñados, generar confianza, además, son realizadas de forma aislada del resto del sistema

(ISTQB, 2018, p.42), y son realizadas por los propios desarrolladores, estas pruebas deben de ser ejecutadas, ya que para que el software, sea probado por nuestra área nos deben de dar evidencia de que fueron ejecutadas con éxito para que pueda pasar hacer probada por mi área.

- Las pruebas de integración es el proceso de verificar la interacción entre componentes o sistemas de software (ISTQB, 2018, p.43), en donde debemos validar que funcionen cuando trabajan en conjunto ya que puede que los componentes o sistemas funcionen de forma individual, pero no cuanto se interrelacionan lo que puede provocar que se presenten errores, por ello que cuando se ejecutan este tipo de pruebas se deben seleccionar flujos de punta a punta validado esta interacción y son ejecutadas por el equipo de desarrolló.

- Pruebas de sistema se centran en el comportamiento y capacidades de todo un sistema o producto, a menudo teniendo en cuenta las tareas extremo a extremo que el sistema pueda realizar. Algunos ejemplos de productos de trabajo que pueden ser utilizados como base de prueba son basadas en análisis de riesgos, especificaciones de requisitos del sistema o *software*, épicas o historias de usuario, casos de uso, modelos de comportamiento del sistema, diagramas de estado, manuales del sistema y del usuario. (ISTQB, 2018, p.46).

  Las pruebas de sistema deben centrarse en el comportamiento global de extremo a extremo del sistema en su conjunto, tanto funcional como no funcional y las características de calidad (ISTQB, 2018, p.47). Son el tipo de pruebas que ejecuto en mí día a día, ya que dentro de esta categoría se encuentran las pruebas funcionales que es precisamente el momento en el que validamos el correcto funcionamiento de la aplicación basándonos en nuestros diseños de matriz de pruebas.

En donde validamos el correcto comportamiento del sistema, y como área de pruebas es donde debemos de encontrar la mayor cantidad de defectos, aunque existen niveles de pruebas anteriores como unitarias y de integración en donde se debieron de encontrar esos issus, aunque no siempre es así, ya que muchas veces estas pruebas no son realizadas y el equipo de desarrollo libera directamente al área testing, las pruebas de sistema son consideradas también como las apropiadas para comparar el sistema con los requisitos no funcionales del sistema, como seguridad, rendimiento, exactitud, velocidad y confiabilidad (SWEBOK, 2004). Este tipo de pruebas son ejecutadas por nuestra área si son solicitadas, siempre y cuando podamos replicarlas en nuestros ambientes de pruebas como por ejemplo cuando tenemos que medir la velocidad que se tiene al cargar archivos con millones de registros o cuando tenemos que validar el rendimiento que se tiene al tener múltiples sesiones a la misma vez, aunque son parte fundamental no siempre están a nuestro alcance, por temas de infraestructura además los ambientes pre productivos no son una réplica exacta de los productivos, por lo que en ocasiones estas pruebas se ejecutan en un nivel posterior UAT (Pruebas de Usuario Final).

- Pruebas de aceptación se centra normalmente en la validación de la idoneidad para el uso del sistema por parte de los usuarios previsto en un entorno operativo o real o simulado, el objetivo principal es crear confianza de que el usuario puede utilizar el sistema para utilizar el sistema para satisfacer sus necesidades, cumplir con los requisitos y realizar los procesos de negocio con el mínimo de dificultad coste y riesgo (ISTQB, 2010, pág. 48). Sin duda es parte crucial debido a que el usuario en esta etapa valida que efectivamente lo que probamos funcione de acuerdo con lo solicitado, mi tarea en este proceso es guiar, orientar y dar seguimiento durante la ejecución de pruebas, además sirvo de soporte cuando se encuentran defectos, sin embargo, mi área no es la encargada de la ejecución si no el usuario.

A continuación se muestra un diagrama en donde se visualiza los niveles de prueba por cada nivel de verificación, la validación se refiere a la corrección de cada nivel de desarrollo, validar es comprobar que los resultados de un nivel de desarrollo sean correctos por ejemplo en las pruebas unitarias se debe validar que sean adecuados los resultados de la generación del código, en las pruebas de integración se debe de valida que se han adecuados el diseño de los componentes, en las pruebas de sistema se debe de validar que sea correcto los resultados del diseño de la arquitectura, en las pruebas de aceptación se debe de validar que se cumplan los requisitos definidos por el cliente como se muestra en la imagen siguiente:

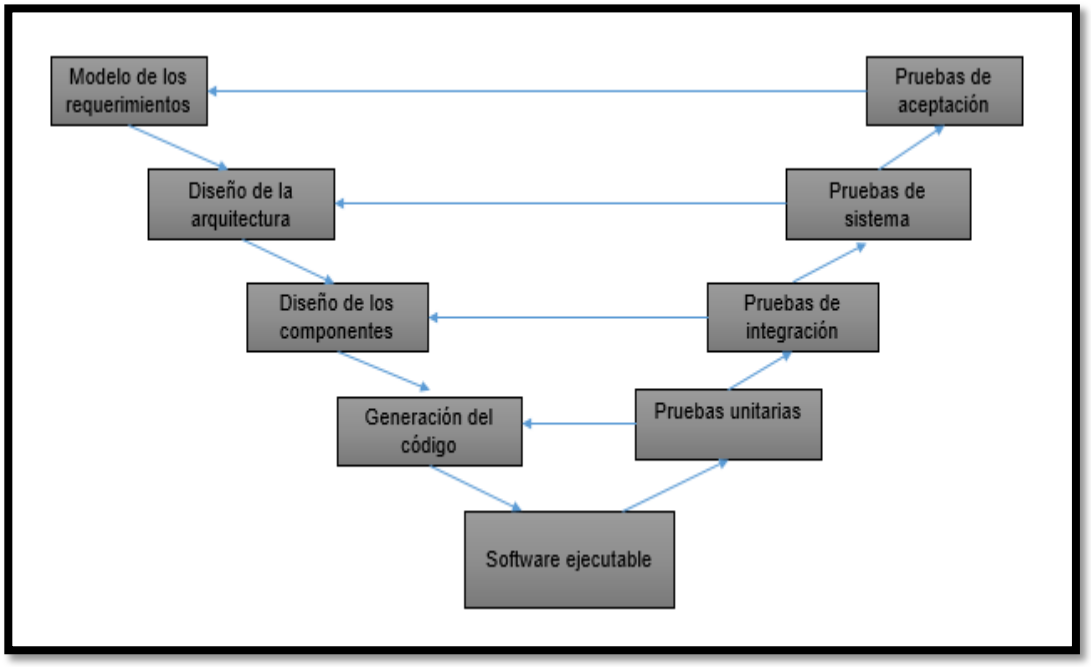

Figura 5. Validación en el modelo de ciclo de vida en V (Roger, 2010).

Como vimos anteriormente el lado izquierdo representa la verificación que se refiere al conjunto de tareas que garantizan que el software implementa correctamente una función específica. La validación es un conjunto diferente de tareas que aseguran que el software que se construye sigue los requerimientos del cliente (Roger, 2010, p.384).

Plan de pruebas

Es importante entender que el diseño de las pruebas no comienza cuando el software construido es puesto en nuestras manos, si no tan pronto los requerimientos son aprobados, por lo que es importante que nuestro líder de pruebas, nos comparta el plan de pruebas ya que nos servirá de guía para la realización de nuestras actividades, ya que es un producto formal que define los objetivos de la prueba de un sistema, se define el hardware, software, permite determinar el calendario y procedimiento de las pruebas, los recursos que se necesitaran para la ejecución, además nos ayuda a obtener un panorama general del proyecto (Sommerville, 2005, p.476)., a continuación, se enlistan los elementos que puede tener el plan de pruebas, sin embargo, más adelante se abordara con más detalle:

- Identificar los componentes de software que serán probados.
- Identificar los tipos de prueba que serán probados.
- Lista de requerimientos de las pruebas.
- Describir las estrategias de pruebas.
- Identificar los recursos requeridos para las pruebas.
- Roles y responsabilidades.
- Listar los productos finales de pruebas (entregables).
- Ciclo de pruebas.
- Manejo de defectos.

Diseño de pruebas

Esta etapa inicia cuando el líder de pruebas me asigna un requerimiento el cual debe haber sido aceptado por el área de gestión de pruebas y asignado a mi equipo, y es en donde comenzamos con el diseño de pruebas que es una de las funciones principales del equipo de Testing, esta actividad puede llegar hacer tan compleja como lo es el desarrolló. En la figura 6 se muestra el proceso de desarrollo de software.

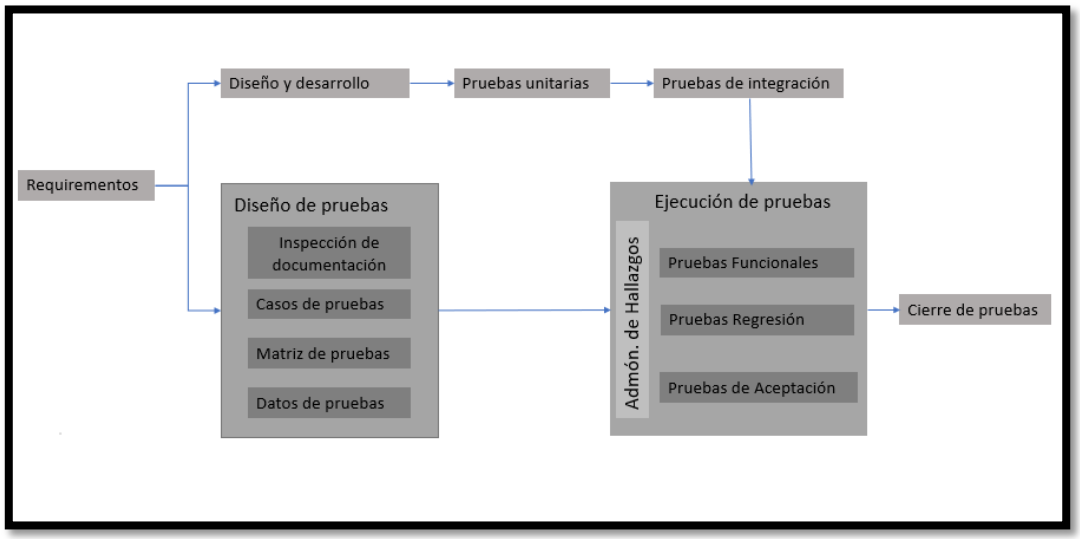

Figura 6. Diseño de pruebas (Creación propia, 2019).

Inspección de documentación

En cuando el Analista comparte la documentación final y me es asignado iniciamos con la inspección, la cual es una técnica estática formal que tiene el propósito de encontrar defectos, sin necesidad de ejecutar el aplicativo / sistema /programa, como por ejemplo casos de uso, documento de requerimientos, documento de reglas de negocio, documento de requerimientos no funcionales, prototipo entre otros., lo que buscamos como primer paso es comprender y entender la funcionalidad de la aplicación, posterior de ello debemos detectar todos los posibles defectos o huecos funcionales, por ejemplo, que en el mismo documento se contradiga, que no sea contemplado algún botón o funcionalidad etc.

En caso de que el proyecto sea complejo, puedo recurrir representar gráficamente la funcionalidad por medio de un diagrama de flujo, que representa una secuencia de pasos necesarios para realizar una tarea mediante símbolos (Pacheco, 2019), el cual creamos a través, de alguna herramienta como lo es .VUE (permite la creación de diagramas) y que me permite tener visibilidad y comprensión de la funcionalidad, los diferentes caminos que podrían llegar a presentar y los pasos a realizar por cada flujo, además, por su diseño es fácil de ser interpretado por otros miembros del equipo. A continuación, pongo como ejemplo en la figura 7, la funcionalidad de *loggin* de un aplicativo y género el diagrama:

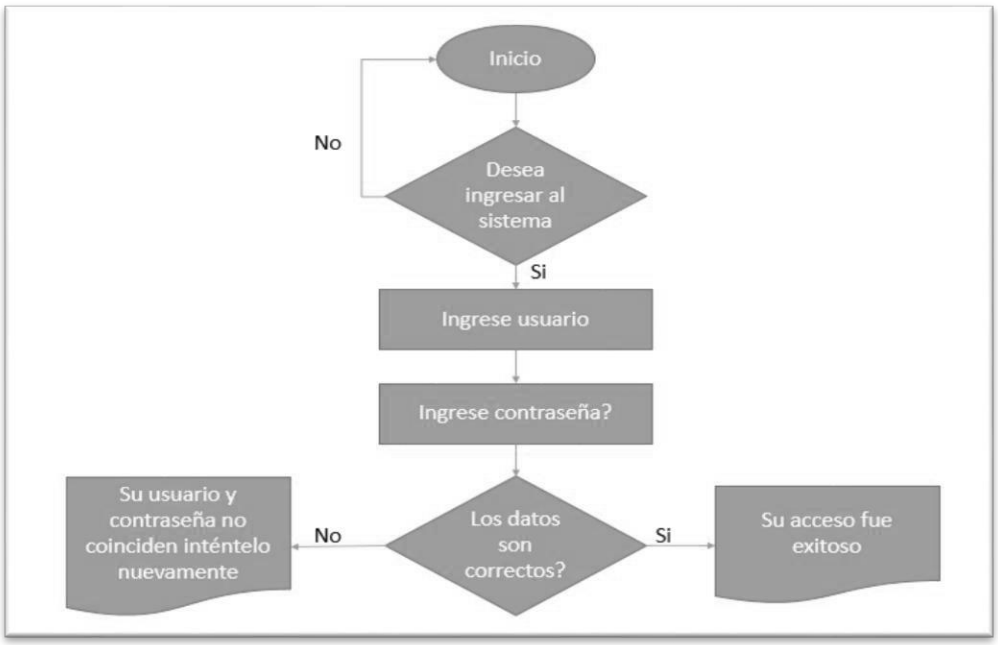

Figura 7. Diagrama de flujo (Creación propia, 2019).

Aunque la funcionalidad del *loggin,* no es complejo me da la visibilidad de la funcionalidad y los caminos que se presentarán, además me ayuda a identificar huecos funcionales, defectos o dudas en la documentación que es precisamente lo que buscamos al realizar la inspección, todas estas dudas deben ser concentradas en un documento o enviadas a todos los involucrados en el proyecto a través de un correo, las cuales deberán ser discutidas en una

reunión con todas las partes involucradas desde Gerente de pruebas, Líder de pruebas, Ingeniero de pruebas, Desarrolladores, Analistas de Negocios, PM (Project Management), Usuario si es requerido etc., está junta debe ser organizada por el PM., y durante la revisión debemos exponer e ir descartando en conjunto todos los falsos positivos (se cree que son defectos, pero en realidad no lo son)., e inclusive podemos llegar a detectar falta de funcionalidad o algún bloqueo para el inicio de pruebas, en caso de que lo reportado aplique debo levantar el defecto, registrarlo en la herramienta de administración, realizar la asignación al analista, que genero la documentación y doy seguimiento hasta su corrección.

Al finalizar la revisión el PM, debe mandar por correo la minuta con todos los acuerdos a los que se llegaron y las tareas, indicando el responsable que las debe de realiza, además me puede servirme de evidencia en un futuro ya que en ocasiones no son respetadas por el equipo.

Casos de pruebas

En cuanto finaliza la revisión y tengo la funcionalidad clara nuestra siguiente tarea es la identificación de casos de prueba donde debemos asignar un nombre corto y claro de la funcionalidad a probar y al ser leído por cualquier persona debe darle visibilidad de la funcionalidad a probar y son obtenidos a partir de la documentación. Para realizar esta actividad, hablaremos de dos herramientas utilizadas en la cuenta asignada.

La primera es la elaboración de diagramas de árbol (es una representación gráfica de una experiencia que consta de múltiples pasos, donde cada uno de ellos posee varias maneras de llevarse a cabo (Pacheco, 2019), su diseño permite obtener la cobertura de pruebas e identificar las trayectorias: *happy path*, flujos alternos, flujos opcionales y excepciones.

Caso de prueba derivado de caso de uso

De acuerdo con ISTQB las pruebas pueden provenir de casos de uso, que son una forma concreta de diseñar interacciones con elementos del *software*, que incorporan requisitos para las funciones del software representadas por los casos de uso. Los casos de uso son asociados con actores (usuarios, hardware externo u otros componentes o sistemas) y sujetos (el componente o sistema al que se aplica el caso de uso) (ISTQB, 2018, pag.76).

El caso de uso sirve para describir el comportamiento de sistema al afrontar una tarea de negocio o requisito del negocio. Esta descripción se enfoca en el valor del suministrado por el sistema a entidades externas tales como usuarios humanos u otros sistemas. Además, promueve una imagen fácil del comportamiento del sistema un entendimiento común entre el cliente/ propietario/usuario y el equipo de desarrollo (Fossati, 2016).

El caso de uso puede incluir distintas variaciones de comportamiento desde el flujo básico que describe la funcionalidad principal del sistema, así como los flujos de excepción que describe los errores que puede presentar el programa cuando se introducen datos que no cumplen con las especificaciones. Las pruebas derivadas de casos de uso están diseñadas para ejercer los comportamientos definidos básicos, excepcionales o alternativos, y manejo de errores (ISTQB, 2018, pag.76). A continuación, se muestra el diagrama de caso de uso que ejemplifica las interacciones que un usuario puede tener con un sistema.

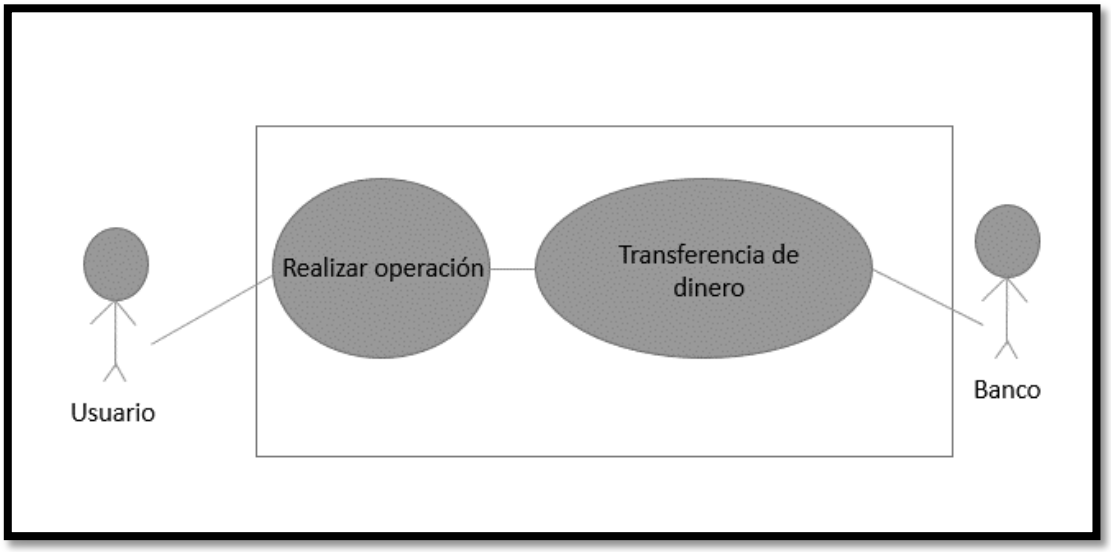

Figura 8. Diagrama caso de uso relaciones (Creación propia, 2019).

A continuación, tomaremos como ejemplo el siguiente documento de caso de uso, para poder ejemplificar el diseño del diagrama de árbol y obtener la cobertura de pruebas.

Tabla 1
Documento del caso de uso "Realizar Transferencia de Dinero"

Datos generales del caso de uso			
Nombre Caso Uso	Realizar Transferencia de Dinero		
Creador	Nombre del autor	**Fecha de Elaboración**	DD/MM/AAAA
Autor que modifica	Nombre del autor	**Fecha de modificación**	DD/MM/AAAA
Objetivo			
Realizar una transferencia bancaria entre dos cuentas bancarias a través del sistema de pagos del portal de banco			

Nivel del Caso de Uso	**Prioridad**	**Complejidad**
Usuario	Prioridad que tiene el caso de uso	Complejidad que tiene el caso de uso

Actores involucrados
Cliente
Evento Disparador
El resultado de la transferencia efectuado por el usuario permite obtener los recibos correspondientes que permiten comprobar que la transacción fue realizada con éxito
Precondiciones
• **Autenticación** El cliente está registrado y tiene autorización para realizar la transacción de transferencia necesaria. Las cuentas están autorizadas para aceptar transacciones desde el sistema de pagos.
Post-condiciones
• Los recibos correspondientes a la transacción son generados de forma exitosa

Flujo básico (Escenario Principal)

El flujo básico permite al usuario realizar una transferencia de fondos bancarios entre dos cuentas

Paso	Acción
1.	El Caso de Uso inicia cuando el usuario autentificado pide una transferencia
2.	El sistema solicita información de la cuenta de destino y la cuenta de origen.
3.	El cliente brinda la información requerida de cuenta de origen y cuenta destino.
4.	El sistema solicita información de la cantidad a transferir
5.	El cliente brinda la información de la cantidad a transferir
6.	El sistema revisa disponibilidad de fondos de la cuenta origen (FA1, F01,EX01)
7.	El sistema solicita autorización al cliente para realizar la transferencia
8.	El cliente confirma la autorización
9.	El sistema contacta a la entidad bancaria de la cuenta destino y realiza la transferencia de fondos
10.	El sistema genera los recibos correspondientes a la transacción
11.	El cliente obtiene el recibo
12.	Fin del caso de uso

Flujos alternos

FA1 – Monto superior al que se tiene en la cuenta

Paso	Acción
1.	El flujo inicia cuando el sistema informa al cliente que el monto solicitado para transferir supera el saldo de la cuenta.
2.	El cliente acepta la notificación y pide ingresar un nuevo saldo.
3	El caso de uso continúa en el paso 4 del escenario principal.

Flujos Opcionales

FO1 –Cancelar transacción

Paso	Acción
1.	El flujo inicia cuando el sistema informa al cliente que el monto solicitado para transferir supera el saldo de la cuenta.
2.	El cliente acepta la notificación y pide no continuar con la transacción
3.	El caso de uso termina.

Flujos de Excepción

EX01 –Sistema no disponible

Paso	Acción
1.	El flujo de excepción inicia cuando el Sistema identifica que no recibe una respuesta exitosa
2.	El sistema cancela la solicitud
3.	El Sistema informa al Solicitante que el servicio no se encuentra disponible
4.	Fin del caso

Nota. Esta tabla muestra como está conformado un documento de caso de uso

El documento de caso de uso nos permite identificar la funcionalidad a probar y tener contexto del negocio, en la tabla 1, se describe el objetivo principal del CU además, permite identificar el flujo principal también conocido como happy path (camino feliz), en donde se describe la función principal del sistema, que como testers debemos de ser capaces de identificar debido a que es el flujo más importante al ejecutar las pruebas y es lo primero que debemos de probar debido a que es la funcionalidad principal, además, en el documento también podemos encontrar los flujos alternos que se desprenden del flujo principal y que son los diferentes caminos que puede tomar la aplicación respecto al flujo principal, en ocasiones también podemos encontrar flujos opcionales que hacen referencia a cuando el sistema puede tomar caminos distintos dependiendo de la acción que ejecuta el usuario, ejemplo al cancelar la transacción la aplicación muestra dos opciones si y no, el flujo opcional, es la opción que no sigue con el flujo principal o alterno, y por ultimo sus excepciones que son utiliza para describir funcionalidades no deseadas a causa de una condición no cumplida, en ocasiones también puede contener requerimientos especiales o también conocidos como requerimientos No funcionales que pueden estar dentro del documento de caso de uso pero que no se especifican en los flujos descritos y son más de rendimiento, usabilidad fiabilidad del sistema etc., ejemplo: Cuantos registros soportara, validación en campos obligatorios etc.

Representación Gráfica

Para aplicar la técnica de creación de *test Cases* partiendo de casos de uso, podemos utilizar varias herramientas como el diagrama de árbol, que es una representación gráfica en forma de grafo, que nos permite tener visibilidad de las distintas rutas que se pueden seguir, obteniendo así la cobertura de pruebas.

Para poder crear el diagrama debemos tener clara la funcionalidad del caso de uso debido que es de donde partiremos para obtener el diseño identificando el flujo principal, flujos alternos de Excepción y opcionales que nos permitirá identificar fácilmente cuales son los flujos que tiene el sistema y los pasos a seguir para cada uno de ellos, tal como se muestra en la figura 9, a continuación, utilizaré esta herramienta para poder ejemplificar como obtenemos las trayectorias y casos de prueba, aplicándola al caso de uso que se mostró anterior mente:

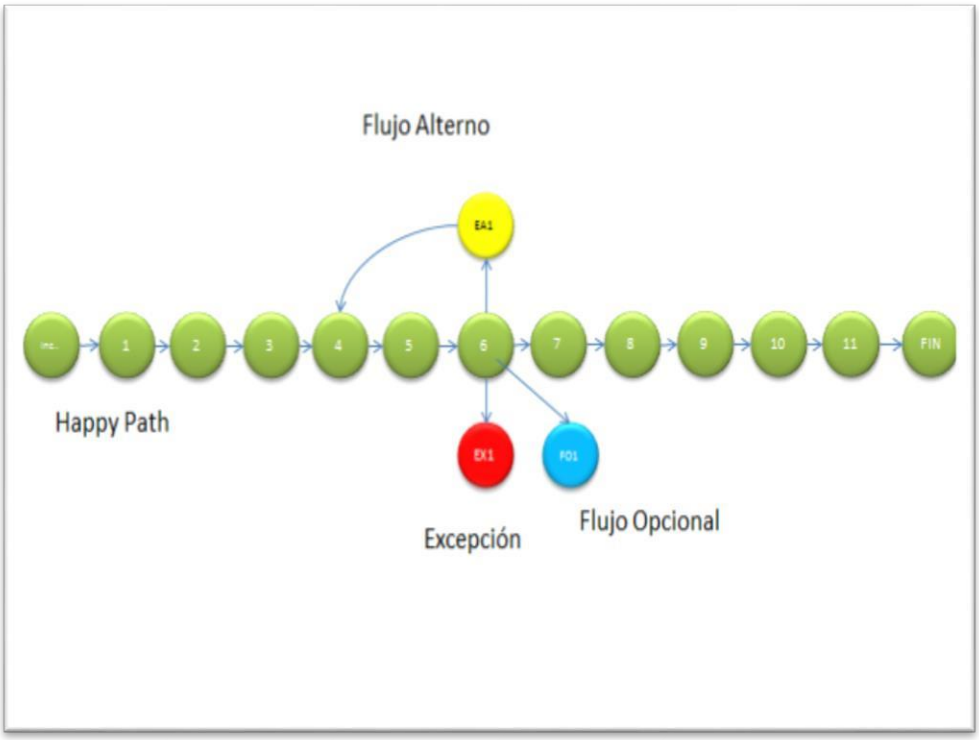

Figura 9. Diagrama de árbol (Creación propia, 2019).

En cuanto se finaliza con el diagrama de árbol trazamos cada una de las rutas o trayectorias a probar en formato ecxel, y posterior debemos dar una descripción breve a los escenarios identificados que deben de ser claros, cortos y describir la funcionalidad a probar, a continuación, se muestra la cobertura casos a probar obtenidos a través del diagrama de árbol elaborado:

	Trayectorias	Descripcion del escenario
1	0-1-2-3-4-5-6-7-8-9-10-11-Fin	Transferencia de fondos bancarios entre dos cuentas de forma exitosa
2	0-1-2-3-4-5-6-EA1	Transferencia de fondos bancarios entre dos cuentas, ingresando un monto superior al que se tiene en la cuenta
3	0-1-2-3-4-5-6-EA1-4-5-6-7-8-9-10-11-Fin	Transferencia de fondos bancarios entre dos cuentas, ingresando un monto superior al que se tiene en la
4	0-1-2-3-4-5-6-FO1	Transferencia de fondos bancarios entre dos cuentas, cancelando la solicitud
5	0-1-2-3-4-5-6-EX1	Transferencia de fondos bancarios entre dos cuentas con el servicio no disponible

Figura 10. Trayectorias (Creación propia, 2019).

Casos de prueba derivados de documento de requerimientos

Los requerimientos del software para el registro de las pruebas se dividirán en requerimientos funcionales y no funcionales. Los requerimientos funcionales son especificaciones de los servicios que debe proporcionar el sistema, de la manera en que este debe comportarse ante entradas particulares de datos y de cómo se debe comportar ante situaciones particulares. Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema (Sommerville, 2005, p 109-113).

Los documentos de requerimientos son utilizamos dentro del proceso de pruebas, nos sirve para entender el negocio y en base a la funcionalidad descrita generar los casos de prueba, que cubrirá con la funcionalidad del software a probar y comienza en cuanto ha sido revisado el *work request* (documento de requerimientos). A continuación, muestro la estructura del documento, tomando como ejemplo la funcionalidad de “compra de artículos en una tienda de ropa online”, que servirá de a poyo para dar un mayor entendimiento:

A continuación, muestro la estructura del documento, tomando como ejemplo la funcionalidad de “compra de artículos en una tienda de ropa online”, que servirá de a poyo para dar un mayor entendimiento:

El documento de requerimientos por lo general contiene los atributos del proyecto, como por ejemplo el nombre del proyecto y su ID, áreas involucradas, sistemas afectados etc., además podemos encontrar el historial de versiona miento que sirve para visualizar los cambios en el documento, proporciona la introducción que tiene como finalidad dar contexto de la funcionalidad a probar y muestra la lista de requerimientos a probar, en ocasiones puede contener pantallas de la aplicación que nos sirven de guía para generar el paso a paso de la matriz de pruebas y por último encontraremos la aceptación y aprobación que desglosa a todas las personas involucradas en la creación y revisión del documento. A continuación, presento un ejemplo de cómo se encuentra estructurado el documento, el cual nos servirá de base para generar los casos de prueba:

Tabla 2
Ejemplo de documento de requerimiento

ID_ Funcionalidad	ID_ Requerimiento	Descripción	Reglas de negocio asociadas	Proceso de negocio afectadas
1	R1	Describe la funcionalidad que el sistema debe ser capaz de realizar como por ejemplo Ingresar a tienda de ropa online y permitir realizar la Compra	Son restricciones en los procesos de negocio como por ejemplo: perfiles específicos, cuenta Registrada	Describe el área que será afectado ejemplo: Ventas
2	R2	Describe la funcionalidad que el sistema debe ser capaz de realizar ejemplo deberá permitir realizar la consulta de los distintos catálogos.	Son restricciones en los procesos de negocio como por ejemplo: perfiles específicos, cuenta Registrada	Describe el área que será afectado ejemplo: Ventas
3	R3	Describe la funcionalidad que el sistema debe ser capaz de realizar ejemplo permite seleccionar y des seleccionar el Articulo	Son restricciones en los procesos de negocio como por ejemplo: perfiles específicos, cuenta Registrada	Describe el área que será afectado ejemplo: Ventas
4	R4	Describe la funcionalidad que el sistema debe ser capaz de realizar como por ejemplo consultar la compra realizada	Son restricciones en los procesos de negocio como por ejemplo: perfiles específicos, cuenta Registrada	Describe el área que será afectado ejemplo: Ventas
5	R5	Describe la funcionalidad que el sistema debe ser capaz de realizar, por ejemplo: Realizar el pago con tarjeta de crédito y depositando en banco (generando ficha de pago)	Son restricciones en los procesos de negocio como por ejemplo: perfiles específicos, cuenta Registrada	Describe el área que será afectado ejemplo: Ventas

Nota. Esta tabla muestra un ejemplo de la estructura de un requerimiento

La generación de casos de prueba inicia cuando ya no tenemos dudas de la funcionalidad, y los vamos identificando con la lectura de los documentos inclusive de prototipos (contiene la funcionalidad y se anexan pantallas del aplicativo), como podemos ver para generar los casos solo nos apoyamos del entendimiento del negocio y de la documentación, ya que por la estructura que tiene es difícil generar un diagrama de árbol.

Los casos de prueba (test cases) identificados deben de tener un formato en específico y debemos colocar el nombre de la aplicación, un identificador y un nombre claro que de visibilidad de la funcionalidad que se probara, además, debemos de asigna el tipo de flujo a probar, a continuación, se muestra un ejemplo:

Tabla 3.
Generación de caso de prueba

Nombre del caso de prueba	**Tipo de flujo**
Tienda online_TC001_Acceso correcto a la tienda de ropa online	Flujo positivo
Tienda online_TC002_Realizar la compra de manera exitosa	Happy Path
Tienda online_TC003_Consultar las distintas categorías en el menú	Flujo positivo
Tienda online_TC004_Permita seleccionar los artículos de la tienda de ropa online	Flujo positivo
Tienda online_TC005_Permita de seleccionar los artículos que no se quieran comprar	Flujo positivo
Tienda online_TC006_Realizar pago por medio de tarjeta de crédito de forma exitosa	Flujo positivo
Tienda online_TC007_Realizar el pago generando ficha de pago para realizar el depósito en banco	Flujo positivo
Tienda online_TC008_Consultar las compras realizadas	Flujo positivo
Tienda online_TC009_Ingresar a la tienda online con un perfil no registrado	Flujo Negativo

Nota. Esta tabla muestra un ejemplo del formato utilizado para la generación de caso de uso

Como se muestra en la tabla anterior el caso de prueba debe de contener una estructura en el nombre empezando por la aplicación, numero de caso de prueba y el nombre que deberá dejar claro el objetivo de la prueba, por último debemos de tener la capacidad de identificar el tipo de flujo aprobar es decir, si es un happy path (describe la funcionalidad principal a probar), flujo positivo (describe los diferentes caminos a seguir) y el flujo negativo (describe las restricciones del sistema, cuando no se cumple con lo especificado).

Dentro de la identificación de caso de prueba podemos recurrir a la técnica de pruebas basadas en la experiencia y como lo indica ISTQB los *test cases* se obtienen a partir de la intuición, competencias que tenemos y también de la interacción que hayamos tenido con aplicaciones similares a las que se probaran (ISTQB,2018).

Es importante saber que, en base a la cobertura de casos de prueba identificados, se determinan los tiempos estimados de pruebas donde se contempla el diseño de los casos, generación de insumos y ejecución de pruebas ejemplo:

Total, de casos a ejecutar = 8

Tiempo invertido para el diseño de caso de prueba = Pongamos 30 minutos si tenemos 8 casos se invertirían 4hrs en esta actividad
Tiempo de ejecución por caso = pongamos 1 hr por cada caso de prueba si tenemos 8 casos se invertirían 8hrs en esta actividad
Generación de insumos = pongamos 20 minutos por cada caso de prueba si tenemos 8 casos se invertirían 2hrs 40 minutos en esta actividad

Tabla 4
Tiempo estimado para la ejecución de casos de prueba

Actividades	Horas	Días
Diseño de casos de prueba	4	4/8 días = 0.5
Ejecución de casos de prueba	8	8/8 días =1
Creación de Insumos	2.4	2.4/8 días=0.3

Nota. Esta tabla muestra el tiempo estimado para la ejecución de casos de prueba

De esta forma obtenemos los tiempos estimados para finalizar con las pruebas del requerimiento asignado, que abarca desde el diseño, creación, ejecución y se agregan días de más por lo regular el doble, para atención de posibles defectos que pudieran llegar a presentarse durante la ejecución o algún inconveniente como retrasos por un ambiente o estable etc.

Cuando tenemos esta tarea completada es importante enviarla a todo el equipo involucrado (desarrolladores, analistas, PM, líder de proyecto etc.), para que den su retroalimentación sobre el estimado, es muy común que soliciten agregar casos de prueba no contemplados o soliciten eliminar casos que no están dentro del alcance de ser así debemos realizar las actualizaciones y enviarla hasta que nos den el visto bueno e inclusive se pueden generar reuniones para explicar cada caso de prueba y clarificar dudas.

Diseño de matriz de pruebas funcionales

Después de tener el visto bueno de los casos a probar, el siguiente paso es crear la matriz funcional que tiene como objetivo recoger los casos de prueba que validaran que el sistema / aplicativo / programa, bajo prueba cumple con la funcionalidad solicitada por el cliente, este documento es importante ya que, en él se describe el paso a paso que el ingeniero de pruebas tendrá que seguir para probar cada test case, en él se debe especificar el resultado esperado de cada caso de prueba a ejecutar, además es donde se plasman los prerrequisitos a cumplir, las precondiciones, post condiciones, parámetros de entrada, evidencias de los resultados obtenidos etc. A continuación, se muestra un ejemplo de la plantilla utilizada para la elaboración del diseño de pruebas:

Matriz de pruebas_Folio- Nombre de Proyecto- Versión

ID_Proyecto	
Nombre del proyecto	
Fecha de creación	
Autor	
Clave de la matriz de pruebas	
Nombre de la funcionalidad a probar	

Nombre de la Aplicación	Tipo de flujo	ID_Caso de Prueba	Nombre del Caso de Prueba	ID_Paso	Descripción Paso a Paso	Resultado esperado	Prioridad	Complejidad	Valores de entrada	Precondión	Post condición	Resultado obtenido	Estado	Evidencia

Figura 11. Plantilla Matriz de pruebas funcionales (Creación propia, 2019).

Este diseño tiene la finalidad de darnos visibilidad de la cantidad de pruebas a realizar derivado del análisis realizado y de la identificación obtenida, además, nos permite establecer la secuencia y relevancia de las pruebas y ver el progreso de ejecución a continuación se describirá el contenido de la plantilla:

Información General

En este apartado debemos describir la información general del caso o requerimiento a probar, estos datos nos permiten que en un futuro podamos recurrir a la persona que creo el documento para que nos dé contexto o nos solvente dudas, además, por la fecha de creación tendremos visibilidad del versiona miento comparándolo con los documentos de requerimientos etc., a continuación, describo cada punto:

- ID_ Proyecto: Colocamos el Identificador del proyecto.
- Nombre del proyecto: Colocamos el nombre del proyecto asignado.
- Fecha de creación: Colocamos la fecha en la que creamos la matriz funcional.
- Autor: Colocamos el nombre del Ingeniero de pruebas que creo la matriz de pruebas.
- Clave de Matriz de pruebas: Colocamos el identificador de la matriz.
- Nombre de la funcionalidad a probar: Describimos el nombre del requerimiento o caso de prueba etc.

Contenido de la plantilla de matriz de pruebas funcionales

El contenido de la plantilla nos permite identificar la aplicación que estará bajo prueba, así como el paso a paso, los datos de entrada, así como el resultado que esperamos y nos da panorama del avance de las pruebas etc., a continuación, describo cada una de las partes que la conforman:

- Nombre del aplicativo: Debemos de colocar el nombre del aplicativo, sistema o programa a probar.

- Tipo de flujo: en este apartado debemos de colocar el tipo ya sea el flujo principal, flujo positivo o un flujo negativo.

- ID Caso de Prueba: Debemos de colocar el identificador único de cada uno de los test case.

- Nombre escenario de prueba: Debemos de colocar el nombre del caso de prueba de manera corta dejando clara la funcionalidad a probar.

- ID Paso: En cada paso que engloba la ejecución del caso de prueba, debemos de colocar un identificador que servirá para obtener el total de pasos que debemos de seguir para ejecutar cada flujo.

- Descripción Paso: Describimos la interacción que el usuario va a tener con el aplicativo para cumplir con el objetivo de la prueba.

- Resultado esperado: Describimos el resultado que esperamos al ejecutar cada uno de los pasos del caso de prueba.

- Prioridad: Debemos de identificar la importancia de la ejecución del *test case* ya sea baja, media o alta y es colocada dependiendo de la importancia que tiene probar el caso de prueba y es dada por nosotros los de QA.

- Complejidad: Debemos identificar y colocar dentro de la matriz la dificultad de la ejecución del caso de prueba sea complejo, medio o simple muchas veces esta es obtenida del número de pasos que tiene el *test case* o del criterio del probador.

- Valores de entrada: Debemos de especificar lo datos de entrada que son requeridos para ejecutar cada caso de prueba: como ejemplo para poder acceder a algún aplicativo los datos de entrada serian: usuario/contra-seña.

- Precondiciones: Debemos de describir lo que es requerido para poder ejecutar el caso de prueba por ejemplo haber iniciado sesión, alguna URL, acceso a BD.

- Post condiciones: Debemos describir los resultados que se deben de cumplir, por ejemplo, persistencia en base de datos, documentos que se deben mostrar en pantalla y guardar en base de datos, cambios de estatus etc.

- Resultado obtenido: Debemos de describir la salida que se espera al ejecutar cada paso del caso de prueba con las entradas indicadas (Datos).

- Estado: Debemos de describir el resultado que se ha obtenido al ejecutar el *test case* ya sea pasado o fallado.

- Evidencia: En este apartado debemos de adjuntar archivos, imágenes o videos que comprueben que el caso fue ejecutado de forma exitosa

Con los puntos tocados anteriormente se realiza el llenado del documento de diseño funcional de la matriz, como ejemplo tomaremos el caso de prueba “Ingresar al aplicativo “X” con éxito, con la finalidad de mostrar esta actividad que realizo en mi área “QA”:

Matriz de pruebas_Folio- Nombre de Proyecto- Versión

ID_Proyecto	
Nombre del proyecto	
Fecha de creación	
Autor	
Clave de la matriz de pruebas	
Nombre de la funcionalidad a probar	

Nombre de la Aplicación	Tipo de flujo	ID_Caso de Prueba	Nombre del Caso de Prueba	ID_Paso	Descripción Paso a Paso	Resultado esperado	Prioridad	Complejidad
Aplicativo X	Flujo Positivo	TC01	Ingresar al aplicativo exitosamente	1	Ingresar url: www.pag.com	Muestra la pagina principal Solicita ingresar *Usuario *Contraseña Muestra botón "Accesar"	Baja	Simple
				2	Ingresar usuario	Muestra el usuario Ingresado		
				3	Ingresar contraseña	Muestra la contraseña ingresada		
				4	Dar clic en accesar	Muestra la pagina de bienvenida		

Figura 12. Matriz de pruebas funcionales paso a paso (Creación propia, 2019).

Matriz de pruebas_Folio- Nombre de Proyecto- Versión

(Ctrl)

ID_Proyecto	
Nombre del proyecto	
Fecha de creación	
Autor	
Clave de la matriz de pruebas	
Nombre de la funcionalidad a probar	

Valores de entrada	Precondión	Post condición	Resultado obtenido	Estado	Evidencia
WWW.Pag.com	Contar con Internet Contar con usuario y contraseña	NA	Muestra la pagina principal	Pasado	Adjuntar evidencia
Usuario	El usuario deberá estar en BD	NA	Muestra el usuario ingresado en pantalla	Pasado	Adjuntar evidencia
Contraseña	El contraseña deberá estar en BD	NA	Muestra la contraseña ingresada en pantalla	Pasado	Adjuntar evidencia
NA	NA	El acceso deberá registrarse en BD	Acceso exitoso	Pasado	Adjuntar evidencia

Figura 13. Matriz de pruebas funcionales paso a paso (Creación propia, 2019).

El ejemplo antes mencionado lo debemos de realizar con cada uno de los casos de prueba identificados y en cuanto concluimos el diseño, lo debemos de enviar al líder de pruebas quien asignara a algún miembro de equipo a realizar el *checklist* (verificar que la matriz cumpla con los puntos establecidos en el documento), a continuación, se muestra la plantilla de ejemplo:

Tabla 5.
Checklist de Matriz de pruebas

No.	Evaluación	Revisión 1	Revisión 2	Se realizó el cambio	¿Es correcto?
1	¿Se cuenta con la información general en la matriz de pruebas?	X	x	Si	Si
2	¿Los casos de prueba cumplen con el formato acordado?	X	x	Si	Si
3	¿Se está respetando la nomenclatura de los casos de prueba?	X	x	Si	Si
4	¿Se encuentran especificados los casos de prueba por complejidad y	X	x	Si	Si
5	¿Están identificadas las precondiciones y post condiciones	X	x	Si	Si
6	¿Están especificados los permisos, roles necesarios para ejecutar los	X	x	Si	Si
7	¿Se tienen identificadas las bases de datos para la validación de	X	x	Si	Si
8	¿Se tiene identificado el resultado esperado de cada caso de prueba?	X	x	Si	Si
9	¿Se están validando la funcionalidad especificada por el usuario?	X	x	Si	
12	¿Se encuentran identificados los datos de entrada?	X	x	Si	Si
13	¿Se está verificando que los datos de salida sean correctos?	X	x	Si	
14	¿Se tiene identificado el flujo principal, flujos positivos y flujos	X	x	Si	Si

Nota. Esta tabla muestra un ejemplo de. Checklist de Matriz de pruebas

El revisor deberá de validar la matriz contra cada pregunta dentro del formato, en caso de que la matriz no cumpla con alguno de los requisitos especificados se nos notificara y tendremos que realizar las correcciones indicadas y volver a enviarla a la persona asignada para su revisión y así hasta que nos dé su aprobación, posteriormente la matriz deberá ser compartida a todo el equipo involucrado y se tendrá que subir a la herramienta utilizada para la gestión de pruebas, también es importante hacer mención que estas revisiones informales las hacemos dentro del equipo es decir, yo puedo hacer el *CheckList* de algún diseño de algún compañero.

En la siguiente imagen muestro un ejemplo, de la herramienta de gestión de pruebas, la cual me permite subir mis casos de prueba y tener un control sobre la ejecución.

Figura 14. Diseño en herramienta de administración de proyectos (Creación propia, 2019).

Ambientación de datos

Una vez que terminamos el diseño e identificamos los parámetros de entrada damos inicio con esta actividad que es importante para poder dar inicio con la ejecución de casos de pruebas, ya que debemos de contar con los insumos (datos de entrada), y deben de cumplir con las condiciones específicas dependiendo el flujo para poder ejecutar la prueba, con la finalidad de validar el comportamiento de la aplicación al ser ejecutada.

Para establecer si el caso de prueba fallo o es exitoso, debemos de identificar las salidas esperadas derivado de los datos de entrada y las condiciones de que establecimos en nuestro diseño

Por ejemplo, si requerimos validar el acceso a alguna aplicación debemos, contar con los siguientes datos antes de que den inicio las pruebas:

- Usuario registrado en BD (Base de Datos)
- Contraseña registrada en BD

Estos serán mis datos de entrada para ingresar a la aplicación, por lo que debo de validar y asegurarme que se encuentren correctamente ambientados revisando que los datos estén registrados en base de datos, mientras que los datos de salida o resultado esperado seria ingresar a la aplicación “X” con éxito esta información debió ser identificado durante la fase de diseño.

Tal como muestro en el ejemplo anterior nuestra tarea es identificar que datos son necesario y que características debe de tener por cada uno de los casos de prueba ya que es importante que contamos con nuestros insumos antes de iniciar las prueba ya que sin ellos no podremos iniciar con la ejecución por lo tanto, debemos anticiparnos y ambientar los datos de acuerdo a lo que identificamos durante el diseño o en su caso solicitarlos al equipo correspondiente y validar que los datos que nos fueron proporcionados cumplan con las características que solicitamos.

Dentro de esta etapa también debemos, de identificar los ambientes de pruebas y solicitar nuestros accesos, si no conocemos las url del aplicativo debemos, de investigarlas con los equipos de desarrollo o responsables, validamos los accesos necesarios y si nos hacen falta debemos solicitar al equipo de datos, identificamos las BD que está relacionada a la aplicación para nuestras validaciones de persistencia, debemos de investigar si se necesita algún rol en específico dentro de la aplicación o si se requerimos registrar alguna url dentro del host, validamos la compatibilidad de las aplicaciones con los distintos exploradores, identificamos si la prueba se realizara atreves de algún *web service* por lo general utilizamos la aplicación SoapUI (herramienta para realización de pruebas), y solicitamos al equipo de desarrollo el *WSDL* (*Web Services Description Language*), *Request* (Petición), con los datos que necesitaremos modificar y también solicitamos el *Response* (Respuesta), con un ejemplo del resultado esperado etc., es importante que contemos con toda la información antes del inicio de la ejecución puesto que, es importante conocer el funcionamiento de la aplicación e inspeccionarlo antes del inicio, si se detecta que en el prototipo no vienen pantallas del *from* (interacción con la aplicación) debemos, de investigar con los analistas responsables.

Los datos los podemos obtener de las siguientes manera: tenemos que realizar la solicitud de insumos con las características específicas requeridas, así como los usuarios registrados en las distintas aplicaciones y base de datos, para ello se debemos de enviar un documento por medio de correo electrónico o atreves de la herramienta de solicitud de insumos, al equipo correspondiente (equipo de datos que se encuentra dentro de mi área) quienes deberá proporcionar los datos de prueba, a continuación muestro un ejemplo de la plantilla para solicitud de datos, sin embargo, este proceso dependerá de la cuenta asignada:

Tabla 6.
Solicitud de datos

Ambiente de Pruebas	Cantidad	Estatus	Tipo de dato	Características
A	5	Activos	Teléfonos	Línea activa

Nota. Esta tabla muestra un ejemplo para realizar la solicitud de datos, que deberá ser enviada por correo.

- En caso de que tengamos acceso a las bases de datos podemos sacar directamente la información, creando las consultas necesarias sin la necesidad de realizar la petición al equipo o inclusive podemos realizar *insert* o *update o delete* para la ambientación de usuarios siempre y cuando tengamos permisos de administrador.

- Para el tema de accesos y urls de pruebas debemos de solicitarlo a los equipos de desarrollo, así como las conexiones a las bases de datos, y si la prueba las realizamos por medio de *web services* es necesario obtener el WSDL y el *responce* y *request* además antes de iniciar la ejecución se debe de tener el ambiente tanto el software instalado, si es en web se debe de contar con las urls así como hardware ejemplo: impresoras, detector de huella, certificadoras, cajas, celulares o tabletas si la prueba es desde la APP (programa que se instala en un dispositivo móvil).

Ejecución de pruebas

Esta etapa es importante porque es donde demostramos si la aplicación construida realiza las funciones establecidas con respecto a los requerimientos y cumple con la calidad esperada por el usuario recordemos que él, la mide con el número de defectos que encuentre o dependiendo de la expectativa que él tenga respecto al producto o la necesidad, podría darse el caso que califique la calidad deficiente cuando lo construido no es lo que esperaba por eso es

importante que como ingeniero de prueba aseguremos que lo construido es lo que esperaba el cliente y al entregar el producto final tenga la calidad que esperada. La iniciamos en cuanto el equipo de desarrollo nos entrega la aplicación/sistema/programa instalado en nuestros ambientes de prueba (entorno en el que se ejecutan nuestras pruebas, la implementación debe ser la apropiada y debe ser lo más parecida a producción).

Es aquí donde ejecutamos nuestra cobertura de pruebas y validamos que se cumplan con los resultados esperados, sin embargo en estas validaciones podríamos encontrar defectos al ejecutar los caso de prueba, dependiendo de ello damos un veredicto siendo una prueba exitosa si cumple con la funcionalidad esperada o no exitoso que indica que durante la ejecución se detectó un defecto debido a que no cumple con el resultado esperado, sin embargo, antes de calificar el TC es importante analizar los resultados contra los datos de entrada por que puede darse el caso de que se presenten falsos positivos (La prueba falla, sin embargo, puede ser por los datos de entrada, por temas de ambiente, pruebas mal ejecutadas etc., lo que puede llevarnos a reportar el defecto cuando en realidad no lo es), o falsos negativos es decir, pruebas en donde no detectamos defectos cundo debimos de haberlos encontrado, es por ello que antes que iniciemos nuestras pruebas debemos tener el set de datos correctamente ambientados, a continuación se muestra un diagrama en el cual se ejemplifica el proceso:

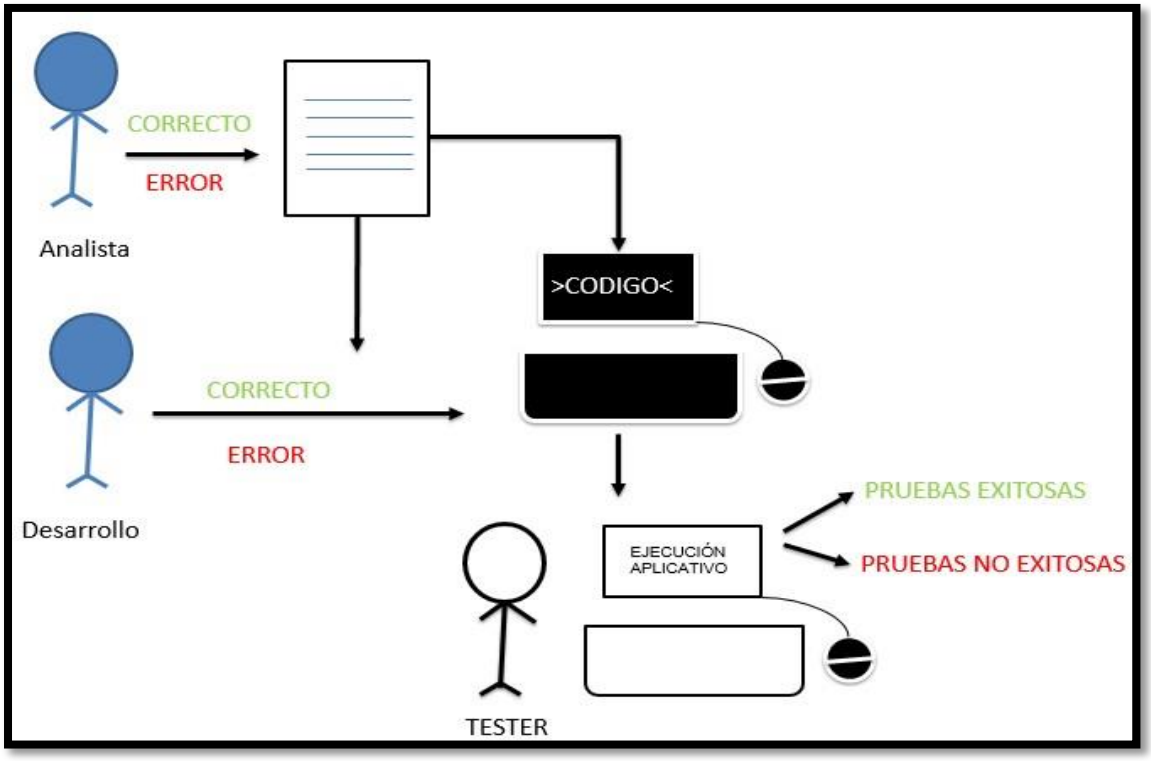

Figura 14. Ejecución de pruebas (Creación propia, 2019).

En la imagen anterior podemos observar que todo parte de la documentación y del entendimiento del analista, si en ella se inyecta un defecto el resto de lo que se realice estará mal, debido a que desarrollo codifica de acuerdo a los documentos (requerimientos, prototipos, reglas de negocio, casos de uso etc.), si durante la creación se cometió algún error por mal entendimiento, distracción, falta de contexto etc., cuando se codifica este error seguirá presente e inclusive pueden cometerse nuevos errores en esta etapa, nuestra labor es precisamente encontrar estos defectos que van surgiendo o quedan ocultos si no se encontraron durante las pruebas estáticas (no se requiere ejecuta el sistema un ejemplo de ello es la revisión de documentación), debemos de ser capaces de detectarlos durante las pruebas dinámicas (se requiere de la ejecución del código para poder probarlo), ya que durante la ejecución somos responsables de validar los resultados obtenidos y dar un veredicto indicando y documentando si la prueba es exitosa, es decir que el resultado obtenido es el esperado o es una prueba fallida ósea que el resultado esperado no es el que se tenía previsto en los documentos ni en nuestros diseños.

También es importante que el desarrollador no ejecute las pruebas funcionales ya que estas deben de ser probada por una persona externa, es decir, si la empresa es la encargada de escribir el código debe contratar a una consultora externa que realice este proceso y debemos estar capacitado para poder hacerlo.

Testing se definen como la ejecución de programas de software con el objetivo de detectar defectos o fallas, aunque tenemos que ser conscientes que no podemos demostrar que un aplicativo no contiene errores ya que aun cuando se demuestre que el aplicativo hace lo que debe, aun así, puede contenerlos de manera oculta. (Pantaleo, 2016).

Como se mencionó anteriormente el proceso para poder dar inicio con la ejecución además, de la entrega de la instalación es que contemos con la matriz de pruebas ya que es nuestra base y se centra en la documentación cerrada, y en ella se encuentra el paso a paso y resultado esperado, por cada uno de los test case, además especificamos las características de los datos de entrada que serán necesarios para la ejecución y nos servirá de base cuando califiquemos el resultado respecto a la funcionalidad esperada de cada uno de los casos de prueba plasmados en ella, a continuación muestro una imagen que representa el resultado de las pruebas.

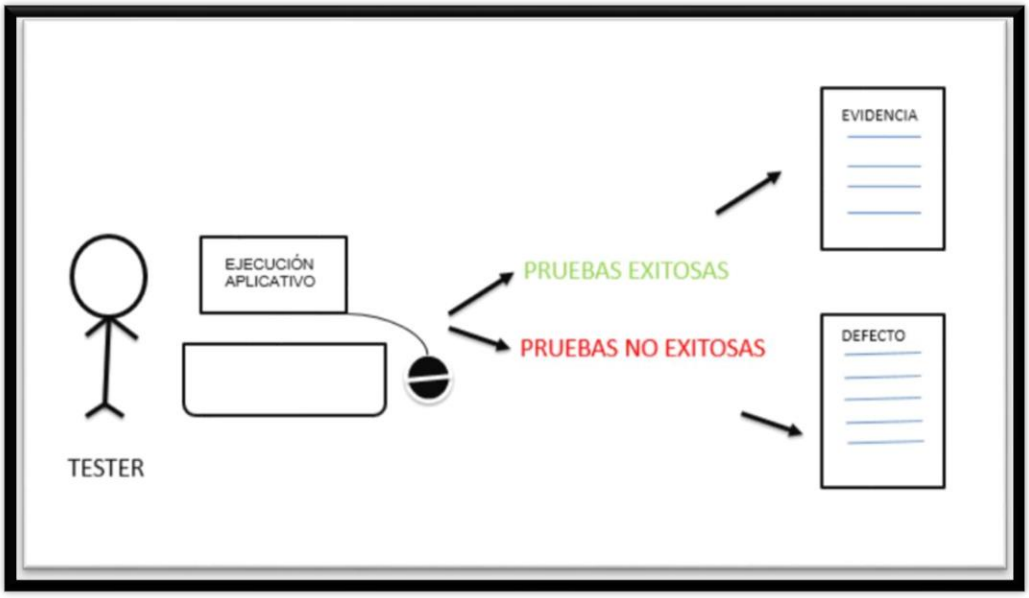

Figura 15. Resultado de pruebas (Creación propia, 2019).

Para ejemplificar este proceso tomaremos como ejemplo el caso de prueba “TC01_Ingresar al aplicativo exitosamente”, tomar en cuenta que para ejecución se deben de priorizar los casos de prueba, es decir, se deberán de ejecutar primero los de prioridad alta, posteriormente los media, y por último los baja y se deberán seguir los pasos descritos en la matriz, en este ejemplo tomo un caso de prueba prioridad baja.

Nombre del Caso de Prueba	ID_Paso	Descripción Paso a Paso	Resultado esperado	Prioridad	Complejidad
Ingresar al aplicativo exitosamente	1	Ingresar url: www.pag.com	Muestra la pagina principal Solicita ingresar *Usuario *Contraseña Muestra botón "Accesar"	Baja	Simple
	2	Ingresar usuario	Muestra el usuario Ingresado		
	3	Ingresar contraseña	Muestra la contraseña ingresada		
	4	Dar clic en accesar	Muestra la pagina de bienvenida		

Figura 16. Ejecución paso a paso parte 1 (Creación propia, 2019).

ID_Paso	Valores de entrada	Precondión	Post condición	Resultado obtenido	Estado	Evidencia
1	WWW.Pag.com	Contar con Internet Contar con usuario y contraseña	NA	Muestra la pagina principal	Pasado	Adjuntar evidencia
2	Usuario	El usuario deberá estar en BD	NA	Muestra el usuario ingresado en pantalla	Pasado	Adjuntar evidencia
3	Contraseña	El contraseña deberá estar en BD	NA	Muestra la contraseña ingresada en pantalla	Pasado	Adjuntar evidencia
4	NA	NA	El acceso deberá registrarse en BD	Acceso exitoso	Pasado	Adjuntar evidencia

Figura 17. Ejecución paso a paso parte 2 (Creación propia, 2019).

Para que probemos nuestro primer caso de prueba tomamos nuestro diseño y ejecutamos cada paso que en él se indique, por ejemplo, para poder realizar la ejecución debo tener mi dato de entrada, el primer paso que tendré que realizar es ingresar a la aplicación y mi resultado esperado es que muestre la página principal, si el aplicativo lo realiza el paso lo validare como pasado y colocamos en ella el resultado obtenido y la evidencia.

Figura 18. Paso 1 Pantalla principal, loggin (Creación propia, 2019).

El paso dos y tres de la matriz indica que se deberá de ingresar el usuario y contraseña y como precondición deberá de estar registrado en BD, al ejecutar el paso validare el resultado esperado que me indica que se deberá demostrar en pantalla, si esto es correcto lo validare como pasado.

Figura 19. Paso 2 y 3 *Loggin* de la aplicación (Creación propia, 2019).

Y el último paso indica que se deberá dar clic en el botón “accesar” y como precondición se debe validar que se registre en BD y el resulta esperado a validar es que nos muestre la página de bienvenida, si la aplicación lo realiza será dado como pasado y se registrara en la matriz el resultado obtenido y la evidencia.

Figura 20. Página de bienvenida (Creación propia, 2019).

En cuanto se finaliza con la prueba debemos de calificar el *test case* de acuerdo a la documentación y podrá ser:

Un resultado exitoso si cumple con cada uno de los pasos descritos y el resultado obtenido es el esperado de acuerdo con la documentación, el cual deberá de ser documentado con imágenes o con un video dependiendo el tipo de evidencia que se solicite agregar anexándolas en la Matrix de pruebas o en la herramienta de administración de pruebas solicitada al ejecutar el registro de ejecución, esta evidencia puede servir en el futuro si se llegara a presentar fallas en producción o si alguien en algún momento requiere ejecutar la prueba.

Un resultado fallado o no exitoso significa que en alguno de los pasos descritos el resultado obtenido no es el que esperábamos, sin embargo, antes de que reportemos el error debemos de validar todos los posibles escenarios por los que el resultado haya fallado ejemplos: validamos que el ambiente se encuentre estable, que el desarrollo se encuentre instalado en los ambientes de prueba, validamos que lo insumos estén correctamente ambientados etc., tomando el caso de prueba ejemplificado podría darse el caso que el aplicativo no nos permita ingresar, en este escenario se debe de validar que efectivamente el usuario y contraseña se encuentren registrados en las bases de datos correspondientes para que al momento de ingresar permita el acceso es importante validar todos los escenarios posibles y en caso de que todo lo encontremos correcto y el *issu* persista se deberá levantar el defecto en la herramienta agregando evidencia y reportarlo al área correspondiente.

A continuación, mencionaré algunos tipos de prueba que eh ejecutado durante mi estancia en las cuentas asignadas en la empresa, lo que me ha permitido ampliar mi conocimiento en esta área, si bien las pruebas funcionales de las que eh hecho mención las ejecuto todos los días, existe una infinidad de pruebas que se pueden ejecutar que a continuación mencionare:

Pruebas no funcionales: existen una infinidad de pruebas relacionados a esta categoría, por ejemplo, fiabilidad, eficiencia, usabilidad, mantenibilidad y portabilidad y se centran en aspectos muy importantes del comportamiento del producto, pero estas no están relacionadas con las funciones que realiza el sistema a continuación se ara mención de pruebas que se pueden llegar a realizar si es solicitado por el equipo.

Pruebas de estrés enfocado a detectar los límites de datos o usuarios que puede soportar al estar conectados simultáneamente, como por ejemplo archivos de datos etc., que en cierto plazo causara que el sistema deje de trabajar o degrade su operación, este tipo de prueba es realizada para detectar la falla y corregirlo, ejemplo de este tipo de prueba supongamos que tenemos un sistema en el cual debemos adjuntar la información de sus clientes a través de un archivo y por cada línea ingresada en el representa un cliente, la prueba de estrés a realizar consistiría en adjuntar tantas líneas como sea posible con el fin de romper la aplicación en esta prueba también podríamos validar el rendimiento de la aplicación es decir, que tan veloz es la respuesta de la aplicación al ser sometida a cierta cantidad de carga, el objetivo de esta prueba es someter al sistema a grandes volúmenes de datos para determinar si el mismo puede manejar el volumen de datos especificados.

Pruebas de regresión: Este tipo de prueba brinda la seguridad de que la aplicación funciona como a lo esperado, es ejecutada debido a cambios en el código que es liberado a producción por cambios en el. Y las debemos de realizan por flujos completos, es decir, ejecutar nuestras pruebas con las diferentes alternativas que se tienen y seleccionando los flujos más importantes y representativos y podemos incluir pruebas de los cambios que se realizaron, porque cuando el código sufre cambios, debemos asegurarnos de que conecta sin ningún problema con el entorno en donde va a vivir y que tenga el comportamiento esperado estas pruebas son funcionales sin embargo en ocasiones son ejecutadas por un equipo distinto.

Pruebas automatizadas: Estas pruebas son ejecutadas en la máquina virtual a través de un robot, el cual tiene cargado los flujos E2E que se han desarrollado y están en producción, por lo general son ejecutadas solo para la generación de insumos y disminución en los tiempos de creación, con el fin de que nos permita tener más productividad en otras actividades.

Pruebas UAT (Pruebas de Aceptación de Usuario): Este tipo de prueba engloba otra área dentro de QA, sin embargo, dependiendo la cuenta asignada puede ser ejecutadas por el tester o por el equipo UAT que en si es un soporte técnico al usuario en donde resolvemos sus dudas, generamos sus insumos, levantamos sus defectos y les damos seguimiento y calificamos sus resultados por cada prueba que el usuario realice e inician posteriores de haber finalizado con las pruebas QA.

Para que puedan ejecutar la prueba el usuario debe seleccionar un set de pruebas derivado de las pruebas funcionales, o bien, el usuario puede generar su propia matriz a validar y se enfocan en validar si el sistema es “apto para el uso”, el usuario es quien se encarga de realizar estas pruebas, y validar que el sistema realice lo que el pidió, esta es la atapa más importante ya que es en donde se gana la confianza del cliente y se muestra la calidad de las pruebas.

El ambiente de ejecución debe de ser lo más parecido a producción y del visto bueno que de los usuarios se determina si las pruebas serán liberadas a producción. Sin embargo, para que se pueda cumplir, deben presentarse muy pocas o ninguna incidencia (bug) durante las pruebas de aceptación, pues de no ser así, lejos de crear confianza más bien podría tener el efecto contrario además pueden servir para evaluar el grado en que el sistema está listo para ser implementado.

Elaboración de consultas en BD (Base de Datos), para la validación de la persistencia

La elaboración de consultas se realiza en SQL (*Structured Query Language*), la finalidad es que validemos la persistencia de información en las tablas correspondiente de cada escenario ejecutado, ya que no tendría sentido ejecutar el caso de prueba validando solo el from (parte visible para el usuario).

Pongamos como ejemplo el siguiente caso: al realizar el registro de un usuario en el aplicativo X, debemos de validar que los usuarios se registre de forma exitosa, el cliente solo tiene acceso al From y los mensajes en pantalla son la única forma de corroborar el resultado, si solo nos basamos en este resultado y no validamos persistencia, puede ocasionar una falla cuando el aplicativo esté en producción, ya que al momento de ingresar con el usuario y contraseña registrada el acceso puede ser denegado, esto se puede evitar si en las validaciones no solo validemos el from si no también la persistencia en base de datos es decir, que los usuarios se guarden en las tablas correspondientes para que permita el acceso de forma correcta.

Levantamiento de defectos

Los defectos son introducidos al software como resultado de un error que se puede dar por mal entendimiento del negocio, distracción etc., y produce un comportamiento incorrecto y no de acuerdo con lo especificado en los requerimientos, a continuación, lo ejemplifico en la siguiente figura:

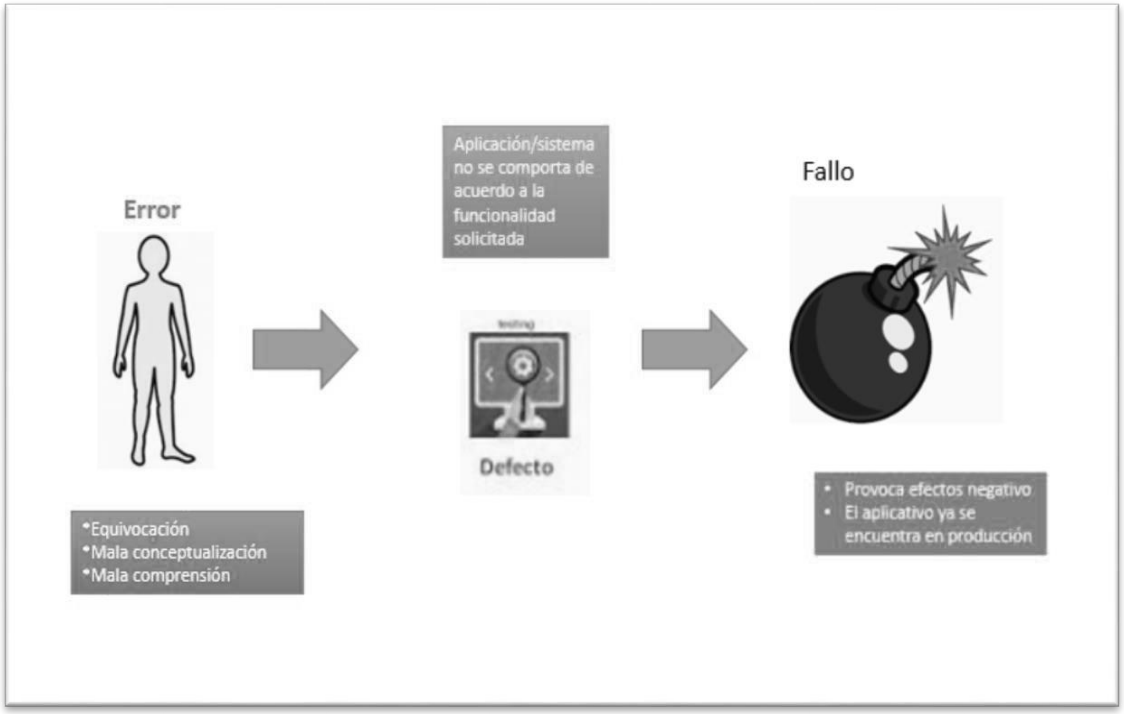

Figura 21. Error, defecto y fallo (Creación propia, 2019).

La detección de defectos es importante porque si los encontramos durante nuestra ejecución, debemos proceder a reportarlos en las herramientas correspondientes anexando evidencia de que no funciona para lo que fue creado y debemos de documentarlo anexando el requerimiento, casos de uso, prototipo, reglas de negocio etc., en donde se describe el funcionamiento correcto del aplicativo asignándolo al responsable del defecto, además debemos de notificarlo a PM, Analistas, Coordinador, líder de pruebas, desarrollo etc., para que esté al tanto de los errores encontrados. A continuación, se describen la información que debe de contener el defecto cuanto los registramos:

Nombre del defecto en donde debemos de describir de manera clara y corta de que trata el *Issu* que nos presentó el aplicativo al ejecutar el caso de prueba con la finalidad de que cuando sea leído por otra persona sepa de qué trata.

- Proyecto de prueba en donde debemos de indicar el nombre de proyecto al que pertenece el caso de prueba ejecutado y que tendrá relación con el defecto encontrado.

- Equipo Asignado en donde debemos de indicar el equipo responsable al que pertenece el defecto, quien será el encargado de realizar las correcciones del defecto y cuando tenga la solución nos debe notificar para realizar el *retest* (volver a probar).

- Propietario del defecto en donde debemos de indicar el nombre de la persona a la que pertenece el defecto encontrado, para que le dé solución, en caso de que el defecto no le pertenezca deberá de asignarlo al equipo correcto indicando el motivo y anexando evidencia para su seguimiento.

Severidad: Debemos de clasificar el tipo de defecto por severidad, es decir, que tanto impacta a la ejecución del aplicativo, se categoriza en criticó, alto, medio y bajo a continuación los describo:

- Crítico: Lo asignamos cuando el defecto no permite finalizar el flujo y afecta el cumplimiento al cliente, además puede provocar pérdidas económicas.
- Alto: Lo asignamos cuando el defecto permite finalizar el flujo, pero no cumple con el requerimiento o reglas de negocio o acepta datos inválidos.
- Medio: Lo asignamos cuando el defecto no interrumpe el flujo debido a que permite finalizar el flujo, pero puede causar confusión o errores humanos.
- Bajo: No detiene al usuario de seguir adelante, corresponde a errores estéticos.

La asignación del estatus nos permite visualizar si nos encontramos bloqueados (crítico, alto), estos impiden continuar con la ejecución de forma total o parcial y deben ser corregidos inmediatamente para no retrasar las pruebas, no bloqueantes (medio, bajo) detectados durante la ejecución, pero que no impiden continuar con las pruebas.

Prioridad hace referencia a que tan rápido requerimos que el defecto sea corregido, basado en la severidad y requerimientos:

- Crítico: debe ser arreglado inmediatamente y no será posible salir a producción, hasta su corrección.
- Alto: debe ser corregido inmediatamente y no es posible liberar la aplicación, pero se puede lograr en el siguiente ciclo.
- Medio: Deberá ser arreglado y se podrá liberar, sin embargo, se deberán realizar las correcciones en el siguiente ciclo si se crea un riesgo significativo.
- Bajo: no se arreglan hasta que los otros defectos sean resueltos. No hay prioridad para arreglarlos.

Descripción de defecto en donde debemos de describir el defecto de forma clara y detallada con la finalidad de que la persona a la cual se le asigna entienda de que trata y pueda dar seguimiento, ejemplo al ingresar a la aplicación X, con el perfil X, al ingresar al menú X y dar clic al apartado X, muestra mensaje de error java.lang, cuando el documento indica que se debe de mostrar el apartado X, debemos describir y detallar el comportamiento obtenido y el proceso que seguimos porque si es ambigua el responsable nos puede rechazar el defecto.

Paso a paso en donde describimos los pasos a seguir para poder replicar el defecto y se deberá indicar en donde está fallando y servirán de apoyo para que el equipo de desarrollo lo pueda replicar sin problema.

Resultado esperado es en donde describimos lo que estamos esperando al realizar la ejecución de la prueba, por ejemplo: visualizar la pantalla de bienvenida y solicite ingresar el usuario y contraseña y el botón aceptar.

Evidencia en donde debemos de adjuntar imágenes, documentación, video y tiene como finalidad evidenciar el comportamiento incorrecto de la aplicación, y que el responsable pueda descargarla y vea el *issu* que se presenta además le dará mayor visibilidad y contexto.

En cuanto se tiene el defecto documentado con todos los pasos anteriores se debe de generar para que se asigne el folio y se pueda dar inicio con el seguimiento a continuación se muestra el ciclo de vida del defecto:

Figura 22. Ciclo de vida del defecto (Creación propia, 2019).

Como muestro en la imagen anterior el seguimiento que se le da a un defecto inicia desde la creación, en la siguiente tabla describo los distintos estatus que podemos tener antes de su solución:

Tabla 7.

Descripción de estatus del defecto

Estatus	Descripción
Abierto	Nos Indica que se ha generado el defecto por parte de QA, por una incorrecta funcionalidad en el software o documentación.
Asignado	Nos indica que ha sido a signado al equipo a cargo de darle seguimiento y quien será responsable de realizar el análisis, esta acción la puede realizar el *tester* o el administrador de defectos.
En progreso	El desarrollador o analista cambia a este estado cuando comienza a trabajar en la solución del defecto.
Listo Para probar	El desarrollador o analista cambia a este estado en cuanto se ha dado solución (Instalación en el entorno de pruebas o corrección de documentos).
Fallido	Cambiamos a este estatus cuando al *retestear* el resultado no es exitoso.
Re abierto	El defecto manager, cambia a este estatus cuando el retest fallo y se continúa su flujo.
Cerrado	El Cambiamos a este estado cuando finaliza la validación y se corrige el defecto obteniendo un resultado exitoso.
Rechazado	El desarrollador o analista cambia a este estado indicando el motivo del rechazo.
Cancelado	Cambiamos a este estado cuando se acepta el rechazo.

Nota. Esta tabla muestra la descripción de los diferentes estatus por los que pasa un defecto

Es importante que demos seguimiento a todos los defectos detectados durante la ejecución de las pruebas hasta que sean solucionados y cerrados o en su caso cancelado anexando evidencia que avale el estatus obtenido finalmente y que quede como respaldo para los siguientes niveles de pruebas.

Informe detallado a los involucrados

Lo reportamos diariamente a los involucrados del requerimiento (PM,S&R, Desarrolladores, Coordinadores, Lideres etc.), en donde debemos poner lo más relevante desde el nombre del proyecto, el porcentaje de ejecución , la información relevante, detalle de la ejecución es decir, muestra el avance de casos de prueba por estatus (pasado, fallado, bloqueado, ejecutado y no ejecutado), defectos levantados por severidad y se indica cuantos defectos han sido cerrados, además dentro de la información relevante debemos detallar los defectos detectados, agregar el equipo, persona asignada y la fecha en la que se le dará solución.

El reporte lo debemos enviar durante todos los días de ejecución hasta que finalicemos las pruebas, en donde informamos a todo el equipo el avance de las pruebas y el estatus en que se encuentra el proyecto siendo en ejecución cuando no existen impedimentos para el avance, fallado cuando existen defectos y bloqueado cuando no se puede avanzar con las pruebas por defectos bloqueantes, además, damos contexto de la información más relevante ocurrida durante el día, como por ejemplo nuevos defectos, si requerimos el apoyo por parte del PM para temas prioritarios, o si identificamos que alguno de los casos quedara fuera del alcance por falta de ambiente etc., además nos sirve para dar contexto al equipo involucrado de nuestro avance y en caso de ser requerido permitirá que el PM nos apoye a escalar algún tema para dar solución y agilizar las pruebas, esto con la finalidad de que se terminen en el tiempo estimado, ver figura 23.

Cierre de pruebas de software y documentación

Da formalidad y pone en evidencia que el proceso de pruebas ha sido culminado, para que pueda ocurrir debemos de ejecutar todos los casos de prueba con éxito, aunque puede darse el caso de que algunos de ellos queden fuera del alcance de ser así, debemos dar justificación y quedar documentado con un correo en donde se dé el visto bueno por parte de los involucrados y que quede como respaldo en caso de que en las etapas siguientes se llegara a presentar un *issu* relacionado, además todos los defectos que hayamos encontrados durante la ejecución deben de estar con un resultado exitoso, y todos los casos de prueba del proyecto deberán de estar ejecutados con el mismo resultado posterior deberemos de enviar el estatus final de nuestras pruebas el cual nos indica que las pruebas han sido ejecutadas al 100% e informamos a todo el equipo involucrado. El estatus puede ser enviado con los siguientes resultados dependiendo las circunstancias en el que finaliza el proyecto y este criterio es dado por nosotros.

- Finalizado con éxito: es decir, que todos los defectos que encontramos fueron resueltos correctamente y todos los casos fueron ejecutados con éxito.

- Finalizado con riesgo: En el trascurso de la ejecución detectamos algún riesgo como por ejemplo problemas de ambiente, no se cuente con datos para ejecutar algún caso, el defecto encontrado no pudo ser re testeado en el ambiente de pruebas y posiblemente el equipo decide probarlo en el siguiente nivel de pruebas para no retrasar tiempos etc.

- Cancelado: Nos indica que, por algún motivo, por ejemplo, el ambiente no homologado, desarrollo incompleto, el ambiente no cuenta con las características para poder realizar la ejecución de las pruebas etc., y por alguna de estas razones se decidió finalizar sin haber concluido las pruebas.

En el reporte debemos describir todos los motivos del estado del proyecto con la finalidad de que el equipo relacionado al proyecto esté al tanto del avance, adjuntando evidencia como referencia si es requerida y colocando los puntos más importantes presentado durante la ejecución, también agregamos las gráficas que dan visibilidad del avance obtenido y de los errores y pruebas no ejecutadas o bloqueadas y las obtenemos de la herramienta de administración de defectos utilizada, de igual se agrega la tabla en donde se encuentran los responsables de cada defecto encontrado y el ETA (tiempo de solución del defecto) que no dieron, a continuación muestro un ejemplo de un estatus de prueba este dependerá del avance que se tenga durante el proyecto:

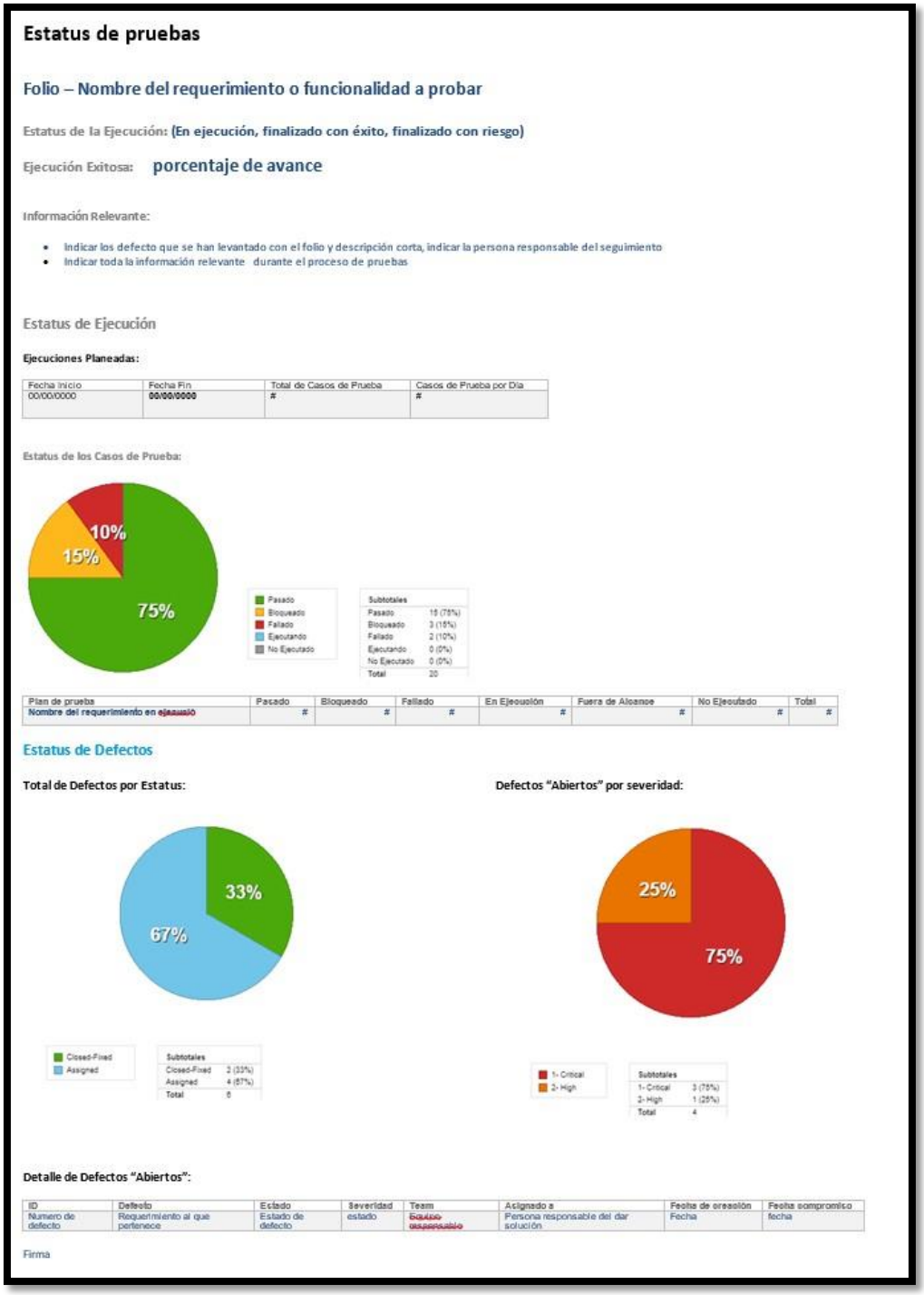

Figura 23. Estatus de Prueba (Creación propia, 2019).

Nuestras pruebas se darán por exitosa, en cuanto se tenga la ejecución de las pruebas al 100%, esto es indicativo de que la funcionalidad probada cumple con los requerimientos del usuario y tiene la calidad esperada, sin embargo, antes de que el aplicativo sea puesto en producción este deberá ser aprobado por el equipo de UAT que es el siguiente nivel de pruebas y tiene la finalidad comprobar si el software está preparado y lo pueden utilizar los usuarios para realizar las funciones y tareas para las que se diseñó.

Documentación de proyectos

Cuando concluimos con las pruebas el siguiente paso o tarea que debemos realizar es llenar el plan de pruebas dentro de la herramienta de gestión de proyectos utilizada en la empresa, a continuación, describo cada uno de los puntos:

1. Introducción del proyecto: debemos una pequeña introducción del requerimiento.
2. Objetivos: Debemos definir los objetivos del plan de pruebas.
3. Propósito del proyecto: debemos definir el propósito del plan de pruebas.
4. Descripción del proyecto: Definimos la descripción del proyecto del plan de pruebas.
5. Tareas generales del proyecto y calendario: Describimos las tareas del plan de pruebas y las fechas en las que se entregaran.
6. Equipo de pruebas: Agregamos el equipo que estuvo a cargo del proyecto de todas las áreas involucradas.
7. Alcance de las pruebas: Agregamos el alcance de las pruebas:
    - Niveles de pruebas: Debemos agregar el nivel de pruebas para el plan de pruebas.

- Tipos de pruebas a ejecutar: Debemos agregar el tipo de pruebas a ejecutar para el plan de pruebas.
- Procesos de negocio afectados: Debemos agregar los procesos de negocio afectados para el plan de pruebas.
- Módulos probados: Debemos de agregar los módulos a probar para el plan de pruebas.
- Requerimientos: Debemos de describir los requerimientos para el plan de pruebas.

8. Riesgos: Debemos de agregar los riesgos para el plan de pruebas.
9. Dependencias para iniciar con la ejecución: Debemos de identificar todas las dependencias para iniciar con el proyecto.
10. Ambiente de pruebas: Debemos de definir el ambiente para el plan de pruebas.
11. Datos de prueba: Debemos de identificar los datos para el plan de pruebas.

Posterior de haber realizado el llenado del plan de pruebas en la herramienta, se deberán de generarlos siguientes documentos y cargarlos en la herramienta de gestión de proyectos:

1. Documentos que generamos durante el Análisis:

    - Documento de riesgo: Contiene todos los riesgos que identificamos, antes y durante la ejecución de pruebas, por ejemplo: tiempos reducidos para la fase de pruebas, ambiente inestable, falta de recursos para la ejecución, cambio frecuente en la definición de requisitos etc.

- Estimación de pruebas: documento que generamos durante el análisis de las pruebas el cual contiene todos los escenarios a probar y el tiempo que es estimado para la ejecución.

2. Dentro del diseño deberemos de generar los siguientes documentos:

    - Matriz de pruebas contiene todos los casos de prueba con el paso a paso descrito y los prerrequisitos que identificamos para la ejecución de las pruebas.

3. Dentro de la ejecución deberemos de generar los siguientes documentos:

    - Evaluación de preparación de pruebas es un documento que evalúa la preparación de las pruebas.

    - Preparar reporte diario de estatus de las pruebas documento que debe de ser enviado desde el inicio de las pruebas hasta su finalización para dar a conocer el avance de las pruebas.

4. Dentro del cierre deberemos de adjuntar los siguientes documentos:

    - Estatus final de pruebas: Reporte final que es enviado para indicar que las pruebas concluyeron.

    - Lecciones aprendidas documento que contiene todos los puntos importantes de la ejecución de las pruebas y lo que aprendimos del proyecto.

En cuanto concluimos con el llenado de la documentación por completo se debe de informar al coordinador de las pruebas para su revisión y visto bueno y de flujo al siguiente nivel de pruebas UAT, cabe destacar que el coordinador es el encargado de asignar algún tester con el fin de dar apoyo al equipo, si se llega a presentar un pre defecto (el error aún no se ha validado en QA), somos los encargados de validar en conjunto con el equipo si aplica o no y validar el comportamiento en nuestros ambientes de QA, si se detecta que el error reportado aplica debemos de levantar el defecto y dar seguimiento hasta que se solucione y notificar al usuario para que realice nuevamente su prueba en teoría somos servimos de apoyo al usuario durante su ejecución.

VI. SOLUCIÓN DESARROLLADA Y SUS ALCANCES

Como comenté anteriormente una de las problemáticas principales se deriva de los cambios que constante mente pide el usuario sobre la documentación cerrada, y esto ocasiona retrabajos de todas las partes del equipo analistas, desarrollo, pruebas etc., lo que es ocasionado ya que, en las metodologías tradicionales, la etapa de pruebas comienza en cuanto las especificaciones de los requisitos han sido cerrados. Lo que ha llevado a que implementemos estrategias de trabajo en donde es fundamental que el equipo de QA esté presente durante todo el proceso, validando las aptitudes en cada una de las etapas del proceso de desarrollo de software y que trabajemos en equipo con todos los miembros involucrados en el proyecto, no solo con nuestra área, esto ha permitido involucrarnos en los cambios que ha tenido el área y demos valor a la calidad que los usuarios esperan al ejecutar las aplicaciones solicitadas.

Actualmente es importante realicemos la validación de los requerimientos en etapas tempranas, es decir, antes de que los documentos son cerrados esto con la finalidad de que encontremos errores u omisiones durante la fase de análisis y demos retroalimentación al área, es importante analicemos los comportamientos que tendrá el aplicativo, ya que debemos anticipar defectos y verificar que el funcionamiento y la usabilidad de una solución sean correctos.

Todos estos defectos encontrados en los documentos de trabajo debemos notificarlos al área de análisis para que el equipo de trabajo realice las correcciones necesarias y en caso de ser requerido ellos deben de acercarse con los usuarios para validar la funcionalidad.

Se realizan conferencias con todas las áreas involucradas en el proyecto para revisión de los documentos analistas, desarrollo, usuario, pruebas con la finalidad de conocer la funcionalidad que es solicitada por el cliente y demos retroalimentación en base a nuestra experiencia, aquí también podemos llegar a encontrar inconsistencias y podemos dar retro entre todas las partes y evitar que persistan errores en la documentación antes de ser cerrada lo que minimiza los errores que puedan llegar a presentarse en otras fases ya que como hemos venido mencionando es mejor encontrar defectos en etapas tempranas y no en producción ya que los costos para ser solucionados son más altos además al tener estas sesiones minimiza los re trabajos que puedan llegar a presentarse.

También se realizan revisiones en conjunto con el equipo de trabajo y se invita a los usuarios para revisar los casos de prueba que ejecutaremos en QA para que se aseguren de la funcionalidad que solicitaron, se cubre con el set de pruebas que les presentamos y en caso de que visualicen que hacen falta validaciones nos las indiquen para que analicemos si se cubren con alguno de nuestros casos de prueba o lo debemos de agregar, además es importante expliquemos la finalidad de cada uno de los caso de pruebas para que de visibilidad de lo que probaremos a todo el equipo y estemos en el mismo entendido, aparte que al estar el usuario en las sesiones nos da visibilidad sobre casos que no habíamos contemplado.

La implementación de reuniones diarias durante la ejecución de pruebas nos ha permitirnos agilizar el avance del proyecto durante el día y que sea más productivo, porqué ha permitido abordemos los puntos más importantes, como por ejemplo los defectos bloqueantes, que son los causantes del poco avance de pruebas dúrate el día y es en donde el PM presta más atención buscando a las área responsables (Desarrollo, Arquitectos, Analistas e inclusive negocio) para dar una solución lo más rápido posible buscando ETAS para cumplir con sus responsabilidades.

La suspensión del proyecto de pruebas es otra medida que adoptamos ya que si durante la ejecución detectamos más defectos, que avance y a eso le sumamos que los días trascurridos superan el tiempo estimado de pruebas o se encuentre por vencer, se decide si el proyecto debe ser suspendido, está decisión se toma en conjunto con el equipo involucrado en una sesión, al ser aprobada se notifica a todo el equipo (Análisis, PM, Desarrollo ETC.), con una minuta para que se tenga de evidencia, y por medio del estatus de prueba se explican los motivos y los acuerdos, esto permite prestar mayor atención a otros proyectos, acelerar los tiempos de respuesta, y minimiza los sobre esfuerzos a proyectos que no están construidos o falta definición ejemplo:

- Cuando un proyecto sobre pasa el estimado para la ejecución y no se tiene avance o es mínimo y se tienen defectos bloqueantes los cuales no se les da solución o se regresan constante mente a diferentes equipos y nadie se hace responsable o no se ha encontrado solución o la solución tardara semanas, es motivo para suspender el proyecto.

Esto permite que los proyectos suspendidos puedan regresar una fase antes en donde el equipo de análisis puede cerrar definiciones y el equipo de desarrollo debe trabajar en ellas, y entregarnos un producto lo más funcional posible a nuestro equipo.

Cuando el proyecto está listo el proceso que se debe seguir para que nuestro equipo pueda tomar el proyecto es el mismo, es decir, se debe realizar la solicitud al área de gestión de pruebas, quienes asignan el requerimiento al equipo de pruebas correspondiente y a su vez el coordinador del equipo asigna al recurso que llevara las pruebas el proyecto se debe de tomar como nuevo, es decir, desde cero sin tomar en cuenta el antecedente.

Implementación de Scrum (Marco de trabajo para el desarrollo ágil de Software).

Además de todo lo que anteriormente mencione actual mente se ha implementado *Scrum* en la cuenta asignada, esto debido a que los tiempos de entrega se extienden y los proyectos tardan mesen en salir a producción, ya que los requerimientos no están definidos al 100% y sufren cambios durante el proceso de ejecución, ya que es en donde se dan cuenta de que falta funcionalidad en ellos., "*scrum* es un marco de referencia para la creación de software y de la entrega en tiempo y de forma sencilla, se basa en construir primeramente la funcionalidad de mayor valor para la empresa" (Dimes, 2015).

De acuerdo a ISTQB cada iteración tiende a ser relativamente corta (ejemplo: horas, días o unas pocas semanas)., y los incrementos de las prestaciones son proporcionalmente pequeñas como unas pocas mejoras y/o dos o tres prestaciones nuevas (ISTQB, 2018, pag.39)., donde se trabaja en interacciones o *Sprints* lo que permite trabajar con funciones prioritarias del software ya sea sencillo o complejo y que aporta valor al dueño de la aplicación, estos *sprints* no deben de durar más de un mes en base a la experiencia se ejecutan de 1 a 2 semanas, salvo que se tenga algún imprevisto puede llevar hasta tres semanas, como por ejemplo la carga de trabajo en otro proyectos, falta de recursos etc., si bien este marco indica que el equipo debe de estar al 100% en el proyecto esto no siempre ocurre, por lo que es importante notificárselo al *Scrum Master* y se llegue aún acuerdo, además con cada sprint finalizado la funcionalidad entregada aumenta y aportar valor es decir, puede ser usada por el usuario, aunque no se tenga terminado todo el proyecto.

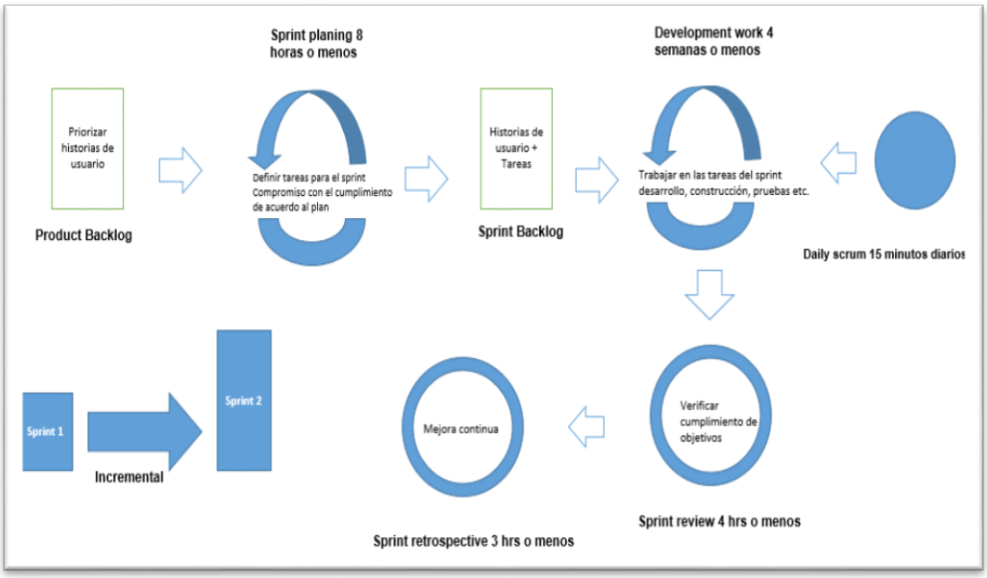

Figura 24. Ciclo de vida de scrum (Ceballos, 2019).

Al trabajar con *Scrum* he podido visualizar que solo existen tres roles importantes y el equipo de trabajo es pequeño, se encuentra conformado por el *Producto owner*, *Scrum master* (facilitador) y el *Team* de desarrollo, es muy importante que como equipo Scrum estemos dedicados por completo al proyecto y participemos en todas las reuniones programadas, por ello cuando nos asigna a un proyecto Scrum, el coordinador de pruebas hace lo posible por no asignarnos ningún otro proyecto, con la finalidad de poder estar dedicados por completo. A continuación, se dará una breve descripción de estos roles:

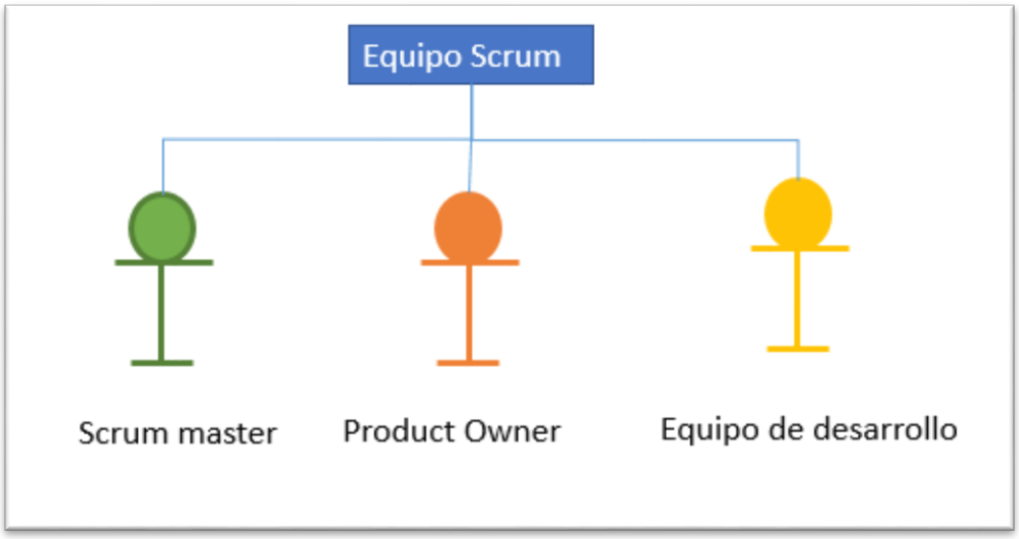

Figura 25. Roles en Scrum (Creación propia, 2020).

Productor Owner: Es la única Voz de los usuarios frente a el *Team,* además es el único responsable del *Product Backlog* en consecuencia tiene la responsabilidad *de* definir historias de usuario y priorizarlas. Además, debe garantizar que las historias cumplan con las necesidades del usuario (Subra y Vannieuwenhuyze, 2018, pag.67).

Scrum master: es el responsable de asegurar que *Scrum* es entendido y adoptado, hacen esto asegurándose de que el equipo *Scrum* trabaja ajustándose a la teoría, prácticas y reglas de *Scrum*, es un líder que está al servicio del Equipo Scrum. El *Scrum Master* ayuda a las personas externas al Equipo *Scrum* a entender qué interacciones pueden ser de ayuda y cuáles no. El *Scrum Master* ayuda a todos a modificar estas interacciones para maximizar el valor creado por el Equipo Scrum (Schwaber y Sutherland, 2013, p.8). Los *Scrum Masters* nos ayuda a entender *scrum* y lo apliquemos de manera correcta, por lo que hace sesiones individuales para ver si conocemos y tenemos clara esta metodología, además nos ayuda a que entendamos la necesidad que se busca en conjunto con el *product owner*, y es el encargado de eliminar todos los obstáculos que se puedan presentar durante los *sprints* como defectos encontrados, falta de tiempo, falta de recursos, falta de definición, que si lo comparamos con una metodología tradicional los tiempos

de respuesta y de solución son más rápidos.

Team Development: El Equipo de Desarrollo consiste en los profesionales que desempeñan el trabajo de entregar un Incremento de producto "Terminado", que potencialmente se pueda poner en producción, al final de cada Sprint. Solo los miembros del Equipo de Desarrollo participan en la creación del Incremento. Los Equipos de Desarrollo son estructurados y empoderados por la organización para organizar y gestionar su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Equipo de Desarrollo (Schwaber y Sutherland, 2013, p.7). Es el equipo encargado de realizar las tareas priorizadas por el *product owner*, es un equipo que debe ser multifuncional y auto organizado además debemos de identificar nuestras tareas de acuerdo con las historias de usuarios que serán ejecutadas en los *sprints*, el *team* está conformado por desarrolladores, analistas y nuestra área entre otras funciones dentro del equipo. Las actividades a realizar en este proceso se describirán a continuación:

Se realiza una primera sesión en donde se presenta el primer artefacto que es el *Product backlog*: en donde se tienen escritas todas las historias de usuario o requisitos/ necesidades del cliente y esta ordenado por valor de mayor a menor, 1 alto, 2 medio y 3 bajo, todos los integrantes debemos de estar presentes y en conjunto se resuelven dudas sobre las historias de usuario y se asignan las prioridades, es parte fundamental ya que si se detecta que falta funcionalidad se agrega, las historias son definidas por el *Product owner,* sin embargo podemos opinar al respecto, además, este rol también se encarga de escribir los criterios de aceptación, conversa y aclara las dudas con el equipo de trabajo y explica cada una de las historias de usuario, si todo es correcto se da el visto bueno, si se necesita alguna modificación se realiza en ese instante, y si por algún motivo no se soluciona se da el tiempo estimado de solución. Además, él debe de ser parte del equipo y tener tiempo para estar involucrado, en todas las sesiones y en todos los *Sprint*, con todos los miembros del equipo,

no es como en las metodologías tradicionales, que solo se relacionan con los analistas, o al menos que surja algún defecto que involucre a negocio se relaciona con el equipo, a continuación, se muestra un ejemplo:

Tabla 8.
Product Backlog

Módulo	**Características**	**Nombre de Historia Usuario**	**Yo como**	**Quiero / necesito Que**	**Para**	**Prioridad**
Actualización de beneficios	Creación de nuevo beneficio tiempo aire	Beneficios Tradicionales	Suscriptor	Contar con servicio sms, llamadas etc.	Poder realizar llamadas y enviar mensajes	1
		Nuevo Beneficio De video	Suscriptor	Poder ver videos	Entretener me, viendo videos con amigos	2
		Paquete de beneficios	Marketing	Proporcionar paquete de beneficios	Para ser más competitivo en el mercado	3

Nota. Esta tabla muestra un ejemplo de Product Backlog

Sprint planing es la primera reunión aquí se planea como se dará solución a la primer fase del producto final, en conjunto se identifican que historias de usuario se probaran durante el sprint y se logra obtener basándose en las prioridades de cada una de ellas, es importante tener en cuenta que solo se deben de seleccionar las historias con las cuales estamos seguros que cumpliremos en esta reunión aun podríamos cambiar las prioridades si así lo requerimos además todos como integrantes del equipo debemos tener claro el objetivo del sprint de hecho al finalizar, el *Scrum Manster* enfatiza y nos realiza esta pregunta ¿Tienen claro el objetivo del Sprint? en teoría al finalizar la reunión tendríamos que tener claro el objetivo del Sprint, las historias que se desarrollaran y los criterios de aceptación, en cuanto tenemos claros estos puntos, debemos de identificar las tareas a realizar, por ejemplo los desarrolladores deben definir qué actividades deben de realizar para lograr con

el objetivo, como equipo de testing debemos identificar las tareas a realizar con el mismo fin, por ejemplo definir el set de pruebas en base a los criterios de aceptación, dar el tiempo estimado para hacer esta actividad en esta parte el equipo podría dar su opinión y en conjunto estimar el esfuerzo e inclusivo detectamos si tendremos participación y se lo indicamos al *Product Owner*, durante esta sesión es importante visualizar los riesgos que pudieran darse durante la ejecución del sprint y las dependencias que podamos llegar a tener, tiene una duración de 8 horas o menos, como resultado se obtiene el 2do artefacto:

Sprint backlog que tiene todas las historias de usuario que se selecciona del *product backlog* más todas las tareas que se detectaron y establecieron en la planeación del sprint aquí es donde definimos si en el sprint habrá pruebas, si es así se indican las tareas que debemos de realizar como por ejemplo la generación de casos de pruebas en base a los criterios de aceptación, los tiempos, el diseño, la generación de datos etc. a continuación se muestra un ejemplo:

Tabla 9.
Sprint backlog

No. Sprint	Nombre Historia de Usuario	Tarea	Horas	Estado	Área responsable Tarea
1	Nuevo beneficio tiempo Aire	Pruebas en laboratorio	96	Pruebas	Ingeniería
1	Nuevo beneficio tiempo Aire	Pruebas en producción	40	Pruebas	Ingeniería
1	Nuevo beneficio tiempo Aire	Documentación	24	En curso	Ingeniería
1	Nuevo beneficio tiempo Aire	Implementar componente x	45	En curso	Desarrollo
1	Nuevo beneficio tiempo Aire	Agregar lógica para implementación de la solución	40	En curso	Desarrollo
1	Nuevo beneficio tiempo Aire	Set de pruebas	8	En curso	Testing
1	Nuevo beneficio tiempo Aire	Ambientación datos	8	En curso	Testing
1	Nuevo beneficio tiempo Aire	Ejecución	8	Por hacer	Testing

Nota. Esta tabla muestra un ejemplo del Sprint backlog

Posterior de haber tenido el *sprint planing*, todo el equipo de trabajo empezamos a desarrollar o ejecutar las tareas que identificamos, aquí es donde intervenimos como *team* de desarrollo y trabajamos en equipo, si bien el equipo de desarrollo inicia con sus actividades, y como testing iniciamos con las nuestras, es importante mencionar que a diferencia de una metodología tradicional en la que realizas un diseño detallado es decir, una matriz de pruebas, con el paso a paso y el resultado esperado y te das a la tarea de investigar cómo funciona el aplicativo, en la metodología *scrum* esto no ocurre por la premura del proyecto, el diseño que se realiza se basa en los criterios de aceptación (descripción del comportamiento esperado del producto) y cada caso se describe de manera muy general sin ningún formato en específico y no se describe el paso a paso, ni el resultado esperado, sin embargo, esto no quiere decir que las pruebas sean ejecutadas incorrectamente, o la forma de diseño no sea correcta ya que esto ocurre debido a que los tiempos en esta metodología son más cortos por lo que debemos ser más agiles. Los escenarios

identificados son revisados en conjunto con todo el equipo para validar que se estará probando la funcionalidad sobre el primer Sprint en donde el equipo puede sugerir agregar o quitar escenarios, a continuación, muestro ejemplo de criterios de aceptación en base a historias de usuario y como esto nos sirve para generar los casos de prueba sin embargo las pruebas en ocasiones se pueden ejecutar solamente con los criterios de aceptación sin la necesidad de sacar los casos de prueba:

		Criterios de Aceptación			
Epica (Módulo)	Nombre corto Historia Usuario	Yo como	Quiero	Para	Criterios de aceptación
Actualización de beneficios	Recibir paquete de beneficio tradicional	Suscriptor	Contar con servicio sms, llamadas	Poder realizar llamadas y enviar mensajes	Al realizar una recarga otorgue mensajes ilimitado y se muestre en pantalla con la etiqueta "Nombre", cantidad y se visualice el consumo,
					Al realizar una recarga otorgue llamada ilimitado y se muestre en pantalla con la etiqueta "Nombre", cantidad y se visualice el consumo
					Al realizar una recarga otorgue Redes socialess ilimitado y se muestre en pantalla con la etiqueta "Nombre", cantidad y se visualice el consumo
					Envie mensaje con el monto de recarga, y los beneficios otorgados

Figura 26. Criterios de aceptación (Creación propia, 2020).

Casos de prueba en base a criterios de aceptación, en base al ejemplo anterior, en donde se describen los criteros de aceptación se inicia con la generación de casos de prueba que se ejecutaran durante el primer Sprint y que deberán de cubrir la funcionalidad a liberar ejemplo:

Casos de prueba	Tipo de flujo
Realizar recarga de $ #, validando que otorgue mensajes ilimitados	Happy path
Realizar recarga de $ #, validando que otorgue llamada ilimitados	Happy path
Realizar recarga de $ #, validando que otorgue redes sociales ilimitados	Happy path
Revisar mensaje por medio de logs validando que se otorguen los beneficios	Flujo positivo

Figura 27. Casos de prueba, (Creación propia, 2020).

Al igual que en las metodologías tradicionales, debemos de identificar nuestros

datos de prueba, si se requiere de algún dato en especial se debe solicitar al equipo de datos, es importante mencionar que con esta metodología los tiempos de entrega se reducen significativamente, además se revisa en conjunto con el equipo de datos y desarrollo, para que se tengan antes del inicio de pruebas, en cuanto a la ejecución la realizamos de la mano con el equipo de desarrollo, ya que las pruebas unitarias, de integración y funcionales se realizan en conjunto, con esto quiero decir que al momento de lanzar una prueba, mientras desarrollo valida el funcionamiento interno por medio de sus logs es decir, que los componentes funcionen de manera correcta, se comuniquen entre sí, etc., nosotros nos enfocamos a validar el correcto funcionamiento del aplicativo en base a las historias de usuario y criterios de aceptación, además validamos la persistencia en base de datos, durante las validaciones se habré un *war room*, con todo el equipo involucrado, lo que facilita la comunicación, si se detecta algún defecto inmediatamente se notifica y desarrollo inicia con la revisión, si no es de su lado, en ese momento se busca y se sube al responsable, hasta la corrección del defecto, o bien, si es una solución compleja se da un tiempo estimado de solución, si se llega a detectar que no es defecto, si no un control de cambio es decir, que el comportamiento es de acuerdo a lo solicitado, pero hubo equivocación u omisión de funcionalidad por parte de negocio, se decide cómo será solucionado, por lo general se van al siguiente *Sprint*, si el defecto es solucionado, inmediatamente nos pide realizar nuevamente la prueba en línea hasta que quede solucionado, esto ha hecho que el levantamiento de defectos sea escaso y el tiempo de solución sea mínimo, y la ejecución avance de manera más rápida, en cuanto a las dudas que surjan durante la ejecución son solventadas de forma inmediata, ya que el *Proyect managment* busca a las personas necesarias de forma inmediata para que sean aclaradas en conjunto.

El segundo evento es el *dayli scrum* con una duración de 15 minutos o menos, el cual es llevado a cabo todos los días y tiene la finalidad de dar seguimiento al sprint y se responden las siguientes preguntas ¿Que hice ayer? ¿Qué voy hacer hoy? y ¿Que me bloquea?, esto para dar seguimiento y tener

contexto del estado actual del sprint lo cual ayuda a la toma de decisiones y a minimizar obstáculos durante el sprint, en esta sesión debemos de exponer todo lo que nos bloquea y no nos permite avanzar con nuestras actividades como por ejemplo falta de insumos para ejecución, también debemos de indicar en que avanzamos un día antes por ejemplo que pruebas ejecutamos, los errores presentados sobre lo ejecutado etc., y que actividades realizaremos durante el día dando una proyección del trabajo que podría ser completado (Schwaber y Sutherland, 2013, p.12).

Spring review es el tercer evento tiene una duración de 4 horas o menos dependiendo de cuantas semas haya durado el sprint, quienes deben participar es todo el team scrum y el Stakeholder (Interesados del proyecto) ya que son los encargados de darnos el *feedback* (retroalimentación) de cómo vamos, en donde se revisa las historias de usuario y objetivos a los que nos comprometimos al inicio del sprint, el dueño de producto explica qué elementos de la lista de producto se han “terminado” y cuales no se han “terminado” y se explica los motivos, después de ello se realiza una demostración del producto o historias de usuario que se desarrollaron, es recomendable proporcionarles herramientas a los Interesados para que ellos ejecuten las pruebas y validen lo que se está entregando y nos den su retroalimentación es decir que les pareció, si ven que requiere algún cambio, si aporta valor etc,. aquí es donde ellos podrían dar el visto bueno o agregar nuevos requerimientos de ser así estos son agregados en el Backlog para contemplarlos en esprints posteriores dependiendo de la prioridad, el Dueño de Producto habla acerca de la Lista de Producto en el estado actual, el grupo completo colabora acerca de qué hacer a continuación, se revisan las historias subsiguientes y se exponen dudas que tengamos directamente con los interesados, se priorizan en conjunto con retroalimentación de los interesados de modo que proporcione información de entrada valiosa para reuniones de planificación de *Sprints* como ejemplo podrían indicarnos historias que no estemos contemplando, aunque en este evento en ocasiones solo se realiza la demostración del producto a los interesados por parte del equipo de desarrollo.

Sprint retrospectivo es llevado por el Scrum Master es otro evento de mejora continua, se analizan los resultados del sprint anterior, ¿Que hicimos bien? por ejemplo trabajar en equipo, disposición por parte del equipo etc., ¿Que hicimos mal? No se le dio el tiempo necesario al sprint, no se cumplieron las tareas en los tiempos estimados etc, y ¿Qué acciones o mejoras vamos a realizar en el siguiente sprint?, por ejemplo, cambiar la dinámica para evitar tiempos muertos, que el equipo se adapte la metodología y los tiempos que esta requiere etc., tiene una duración de 3 horas o menos dependiendo de lo que dura el sprint.

Para el final de la Retrospectiva de Sprint, el Equipo Scrum debería haber identificado mejoras que implementará en el próximo Sprint. El hecho de implementar estas mejoras en el siguiente Sprint constituye la adaptación subsecuente a la inspección del Equipo de Desarrollo a sí mismo. Aunque las mejoras pueden implementarse en cualquier momento, la Retrospectiva de Sprint ofrece un evento dedicado para este fin, enfocado en la inspección y la adaptación (Schwaber y Sutherland, 2013, p.14).

Al finalizar el sprint se inicia uno nuevo tomando otra funcionalidad del *product backlog* y se repite el proceso esto con la finalidad de entregar otro producto funcional y que el usuario pueda ver el avance del proyecto y pueda ir ocupando funcionalidades por cada sprint que finaliza hasta que se tenga la funcionalidad total.

Resultado obtenido con la implementación de este marco de trabajo

1.- Cumplimiento en los tiempos de entrega de la funcionalidad ya que como equipo nos comprometemos a desarrollar solo las historias que consideramos podemos finalizar en el Sprint y esto hace que nos centremos en las historias que serán entregados, si surgen algún problema durante la ejecución de tareas se resuelvan de manera rápida, lo que disminuye la cantidad de errores y permite entregar en el tiempo estimado.

2.- Por cada sprint finalizado y aprobado se aumenta la funcionalidad de los requisitos solicitados por el cliente además puede ser puesta a producción una vez que es aprobada, y utilizada por los usuarios, sin la necesidad de esperar a que el proyecto finalice completamente, ya que esa pequeña parte probada debe ser funcional y aportar valor.

3.- Permite tener reuniones diarias con duración de 15 minutos en donde exponemos lo que nos bloquea, que aremos durante el día y que hicimos un día antes permitiendo que todo el equipo tenga contexto.

Esto nos permite visualizar el avance o la situación en la que nos encontramos, en caso de que se tengan bloques, se dan soluciones para evitar contratiempos y que se implementen de manera inmediata, lo que permite que los tiempos de solución ante errores disminuya de manera considerable, en comparación con una metodología tradicional en donde muchas veces la solución de defectos es solucionada de manera más lenta y se pierde tiempo en la ejecución.

Al implementar esta metodología se evita que se realicen cambios sobre documentos cerrados, ya que el *productor owner* (negocio) es el encargado de escribir las historias de usuario (requerimientos) y explicarnos a todos los miembros del equipo y en conjunto se da el visto bueno o en su caso se deben de realizar las modificaciones necesarias sobre ellas, hasta que todo el equipo

de trabajo esté de acuerdo, esto es importante ya que como equipo nos toman en cuenta, y se da mayor visibilidad, con esto evitamos tener re trabajos durante el desarrollo y ejecución ya que todo los miembros debemos de trabajar en equipo solucionando los problemas que se presente durante los *sprints* para lograr entregar en los tiempos establecidos y cumplir objetivos, también es importante hacer mención que en ocasiones cuando se visualiza que alguna historia contemplada en el sprint no se va a cumplir se pasa al siguiente y esto evita que se tenga retrasos en la entrega.

Además permite que generemos confianza a los clientes al cumplir con los tiempos estimados de entrega y entregando un producto que cumpla con las expectativas de cliente y tenga la calidad que esperan según ISO, define la calidad como “el grado con el que un sistema, satisface las necesidades implícitas y explicitas de sus diferentes usuarios” (Moraga, 2010), el área de pruebas es fundamental ya que contribuye a el logro de niveles apropiados de calidad, aunque es bien sabido que esta se debe de dar en todas las fases de desarrollo de software, sin embargo, nuestra área es la encargada de validar que el sistema funcione de acuerdo a lo solicitado por el usuario y cumpla las expectativas. También es importante hacer mención que, si bien el área de pruebas es conocido en la práctica como área de calidad debido a que debemos de controlarla y mejorarla debemos de ser conscientes que la calidad debe de estar presente en cada una de las etapas del desarrollo del software y no es responsabilidad de un área sino de todas.

VII. IMPACTO DE LA EXPERIENCIA LABORAL

Durante el tiempo laborando en Softtek eh dado calidad en las pruebas realizadas, en cada proyecto asignado, ya que es la tarea más importante de la etapa de pruebas y de proceso de desarrollo de software, debido a que es donde debemos encontrar tantos defectos como sea posible en etapas tempranas ya que es sabido que entre más se tarde en detectar el defecto mayor es el costo, por lo que se le debe de dar continuidad hasta su solución, con la finalidad de entregar un producto que cumpla con cada uno de los requerimientos solicitados por el cliente, para controlar la calidad del software, y se entregue un producto que inspire confianza al usuario y que cuando sea puesto en producción la experiencia del usuario se gratificante, ya que esto hace que crean en nuestro trabajo y calidad y asignen a la cuenta más proyectos, e inclusive recomiende a Softtek y se tenga una buena imagen de la empresa.

El implementar medidas durante los proyectos, permitió reducir los tiempos de entrega e inclusive que algunos de ellos, fueran finalizados en los tiempos estimados, así mismo la implementación de metodologías agiles para algunos proyectos permitió entregar pequeñas funcionalidades del *software*, al finalizar cada *Sprints* en lugar de esperar hasta finalizar las pruebas.

El desempeño en la empresa la productividad y calidad del trabajo en la empresa ha hecho que rápidamente me sean asignados proyectos cuando se culmina alguno y me han dado la confianza de llevarlos de manera independiente, también ha hecho que permanezca en cada una de las cuentas asignadas por Softtek hasta que finalizan los contratos, y que me posicionen rápidamente en alguna otra cuenta, también me ha permitido tener aumentos salariales y subir mi expertice.

El papel de testing juega un papel fundamental en cualquier empresa que se dedique a crear Software, y es nuestra responsabilidad entregar calidad en

el producto final, asegurándonos que cuando sea puesto en producción los errores sean mínimos y que no impacte la funcionalidad, debido a que esto puede generar mala reputación de la empresa.

Las actividades desarrolladas durante más de tres años me permiten poner en práctica los conocimientos adquiridos durante mi formación académica, así como ampliarlo debido a que la empresa cuenta con una universidad en línea que me ha permitido tomar diverso curso, además que la asignación en cada cuenta me ha permitido trabajar de distintas formas y con distintos modelos de desarrollo de software, utilizar nuevas técnica y herramientas y adquirir conocimiento de los distintos negocios.

Mi participación en las cuentas asignadas me ha permitido poner en práctica mi experiencia e ir adquiriendo nuevos conocimientos en el área de como:

- Análisis de requerimientos y casos de uso.
- Estimaciones y generación de casos de prueba con distintas técnicas.
- Diseño de matrices de pruebas con distintas técnicas.
- Ejecución de pruebas.
- Elaboración de consultas.
- Seguimiento de defectos.
- Trabajo con distintos modelos desarrollo de software.
- Utilización de software para generación de reportes, administración de defectos etc.
- Ejecución de pruebas de formas manuales y automatizadas.
- Ejecución de distintos tipos de prueba (Funcionales, UAT, Regresión, Estrés, etc.).
- Generación de reportes
- Elaboración de manuales de usuario
- Capacitación de nuevo personal

Adicional me permitió adquirir habilidades para poder desempeñar la función de manera eficiente y dar calidad en el software o aplicativo, ejecutando pruebas de manera correcta, así como, del servicio que se brinda al cliente:

- Capacidad de trabajar bajo presión.
- Meticuloso (atento a los detalles).
- Capacidad de análisis.
- Pensamiento lógico.
- Habilidad de comunicación oral.
- Capacidad de reportar informes claros y detallados.

De acuerdo con lo vivido puedo decir que es gratificante la labor desempeñada además la empresa se preocupa por cada uno de sus trabajadores proporcionando capacitaciones constantes además el ambiente laboral permite trabajar en equipo y llevar los días de estrés amenos, me ha permitido fortalecer mi experiencia, estar más preparada y capacitada así como enriquecer mi *curriculum* pudiéndome mover a cualquier empresa y desarrollarme en el área de manera sobresaliente, enfrentar nuevos retos que se me presenten durante mi trayectoria laboral.

Gracias a los estudios y la formación en la Universidad Autónoma del Estado de México ya que me preparó para enfrentarme al ambiente laboral y cada materia impartida enriqueció mi formación y conocimiento.

VIII. REFERENCIAS DE CONSULTA

ACM. (2020). Descripción del rol. Recuperado de https://acmgroup.typepad.com/decidimos/files/descripcion_de_rol_lider_funcional_v1.2.pdf

Bogue. R. (2005). Cracking the Code: Breaking Down the Software Development Roles. Development. Recuperado de https://www.developer.com/mgmt/article.php/3490871/Cracking-the-Code-Breaking-Down-the-Software-Development-Roles.htm

Callejas M., Alarcón A., y Álvarez A. (2017). Modelos de calidad del software, un estado del arte. Redalyc, 13(1), 237.Doi: 10.18041/entramado.2017v13n1.25125

Campos. C. (2015). “LAS PRUEBAS EN EL DESARROLLO DE SOFTWARE”. Recuperado de http://132.248.52.100:8080/xmlui/bitstream/handle/132.248.52.100/7627/Las%20pruebas%20en%20el%20desarrollo%20de%20software.pdf?sequence=1

Ceballos Z., (2019).Scrum y metodologías de proyectos. Recuperado de https://xn--zoraidaceballosdemario-4ec.info/scrum/zoraida-ceballos-de-marino-scrum-que-es-y-para-que-sirve-esta-metodologia/

Cessi. (2020). Perfiles desarrollo de software. Recuperado de https://cessi.Org.ar/perfilesit/detalle-de-arquitecto-de-software-3

Dimes (2015). “Conceptos básicos de scrum: Desarrollo de software Agile y manejo de proyectos agiles”. Recuperado de https://books.google.com.mx/books?id=ETuXBgAAQBAJ&pg=PT36&dq=Conceptos+b%C3%A1sicos+de+scrum:+Desarrollo+de+software+Agile+y+manejo+de+proyectos+agiles&hl=es&sa=X&ved=2ahUKEwjW75Tz5orsAhVOKqwKHUwNB5AQ6AEwAHoECAQQAg#v=onepage&q&f=false

Fossati. M. (2016). Testing: Convertite en un experto en QA. Testing: Convertite en un experto en QA. https: // books .google.com.mx /books?id = _kQ1DgAA QB AJ&printsec =frontcover&dq=Testing: +Convertite +e n + un+experto+en+QA&hl=es&sa=X&ved=2ahUKEwik4cP8lYrsAhUH7qwKHU4dAW8QuwUwAHoECAMQBw#v=onepage&q&f=false

IBM. (2017). ¿Qué es DevOps?, Recuperado de https://www.ibm.com/mx-es/cloud/devops?p1=Search&p4=43700052828197561&p5=b&cm_mmc=Search_Google-_-1S_1S-_-LA_MX-_-%2Bdevops_b&cm_ mmca7 =717 00000065289449&cm_mmca8=aud-382859943522:kwd-3196 878 3 0454 &cm_mmca9 =EAIaIQobChMIto2OzrqL7gIV JMmUCR 2V9gMY EAAYASAAEgIpk_D_BwE&cm_mmca10=428815178797&cm_mmca11=b&gclid=EAIaIQobChMIto2OzrqL7gIVJMmUCR2V9gMYEAAYASAAEgIpk_D_BwE&gclsrc=aw.ds

ISO. (2015). Sistemas de gestión de la calidad Fundamentos y vocabulario. Recuperado de https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:es

ISTQB. (2018). Certified Tester Foundation Level Syllabus. Recuperado de https://www.istqb.org/downloads/send/2-foundation-level-documents/281-istqb-ctfl-syllabus-2018-v3-1.html

López y André M. (2006). Roles en el proceso de desarrollo de software para las empresas cubanas. Redalyc. Recuperado de https: //www.redalyc .org/articulo.oa?id=360433560012

Llorens. J. (2005). Gerencia de proyectos de tecnología de información. Recuperado de https://books.google.com.mx/books?id= 7FmOMnfjN_ZIC &pr intsec=frontcover&dq=Gerencia+ de+proyectos+de+t% C3%A9conl ogias+de+inf ormaci%C3%B3n&hl=es&sa=X&ved=2ahUKEwidmo2_12or sAhVDKqwKHWGfCyYQ6AEwAHoECAAQAg#v=onepage&q=Gerencia %20de%20proyectos%20de%20t%C3%A9cnologias%20de%20informaci %C3%B3n&f=false

Moraga. M. A. (2010). Calidad del producto y proceso software. Recuperado de https://books.google.com.mx/books?id=MY0zoXYFVd8C&printsec=frontc over&dq=Calidad+del+producto+y+proceso+software&hl=es&sa=X&ved= 2ahUKEwiMzpb-54rsAhUMiqwKHXrQCFgQ6AEwAHoECAEQAg#v= on e p age&q=Calidad%20del%20producto%20y%20proceso%20softw are_&f =fals

Mera Paz, J. A. (2015). La importancia del proceso de pruebas de calidad del software en la información de los ingenieros en sistemas. Recuperado de https://www.ucc.edu.co/prensa/2015/Paginas/la-importancia-del-proceso-de-pruebas-de-calidad-de-software-en-la-formacion-de-los-ingenieros-de-

sistemas.aspx#:~:text=ser%20Humano%E2%80%8B-,La%20Importancia%20del%20proceso%20de%20pruebas%20de%20Calidad%20de%20Software,los%20Ingenieros%20de%20Sistemas%E2%80%8B&text=Seg%C3%BAn%20la%20IEEE%20el%20proceso,Calidad%20y%20minimiza%20los%20riesgos.

Mera Paz, J. A. (2016). “Análisis del proceso de pruebas de calidad de software”, Ingeniería Solidaria, doi: http://dx.doi.org/10.16925/in.v12i20.1482

Pacheco. J. (2019). ¿Qué es un diagrama de flujo y cómo se hace? Web Empresas. Recuperado de https://www.webyempresas.com/diagrama-de-flujo/

Pacheco. J. (2019). ¿Qué es un diagrama de árbol y para qué se utiliza? Web Empresas. Recuperado de https://www.webyempresas.com/que-es-un-diagrama-de-arbol/

Pantaleo G & Rinaudo L. (2016). Ingeniería del software. México: Alfaomega.

Reqlut. (2018). Líder QA. Recuperado de https://www.reqlut.com/trabajar-en-banco-bci/trabajos/lider-qa-476/16931

Roger. S. (2010). Ingeniería del Software un enfoque práctico. México: MC Graw Hill

Subra y Vannieuwenhuyze. (2018). Scrum, un método ágil para sus proyectos. Recuperado de

https://books.google.com.mx/books?id=TyQuFpGhZ8sC&pg=PT2&dq=scrum&hl=es&sa=X&ved=2ahUKEwjjq-qryYvuAhVBd6wKHVaxBcsQ6AEwAHoECAAQAg#v=onepage&q=scrum&f=false

Schwaber K & Sutherland J. (2013). La guía de Scrum. Recuperado de https://www.scrumguides.org/

Sommerville. I. (2005). Ingeniería del software. Recuperado de https://books.google.com.mx/books?id=gQWd49zSut4C&printsec=frontcover&dq=plan+de+pruebas+en+testing&hl=es&sa=X&ved=2ahUKEwjBg7rRhfnrAhUNna0KHZz5AeIQ6AEwB3oECAgQAg#v=onepage&q&f=false

Softtek. (2019). IT Services and Business Process Solutions. Recuperado de https://www.softtek.com/

Study. (2020). Project Manager vs. Delivery Manager. Recuperado de https://study.com/articles/project_manager_vs_delivery_manager.html

Swebook. (2004). Guía al cuerpo del conocimiento de la ingeniería de software. Recuperado de http://www.cc.uah.es/drg/b/HispaSWEBOK.Borrador. pdf

Tovar B F. (2018). La ingeniería del liderazgo técnico. Asesoftware. Recuperado de https://asesoftware.com/site/ingenieria-liderazgo-tecnico/