UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREAS DE CIENCIAS ECONÓMICAS Y DE GESTIÓN
POSTGRADO EN GERENCIA DE PROYECTOS

PROYECTO DE TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN PLAN DE GESTIÓN DE PRUEBAS PARA LOS PROYECTOS DE SOFTWARE BASADO EN LAS BUENAS PRACTICAS DE GERENCIA DE PROYECTOS.

Presentado por:

Andrade Gil, Tania Josefina

Para optar al título de:
Especialista en Gerencia de Proyectos

Asesor:
Msc. Guillén Guédez Ana Julia

Caracas, Septiembre de 2016

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREAS DE CIENCIAS ECONÓMICAS Y DE GESTIÓN
POSTGRADO EN GERENCIA DE PROYECTOS

PROYECTO DE TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN PLAN DE GESTIÓN DE PRUEBAS PARA LOS PROYECTOS DE SOFTWARE BASADO EN LAS BUENAS PRACTICAS DE GERENCIA DE PROYECTOS.

Presentado por:

Andrade Gil, Tania Josefina

Para optar al título de:
Especialista en Gerencia de Proyectos

Asesor:
Msc. Guillén Guédez Ana Julia

Caracas, Septiembre de 2016

Directora del Programa Gerencia de Proyectos

Estudios de Postgrado

Universidad Católica Andrés Bello (UCAB)

Presente.-

CARTA DE ACEPTACIÓN DEL ASESOR

Tengo a bien dirigirme a usted a fin de informarle que he leído y revisado el borrador final del proyecto de Trabajo Especial de Grado titulado "**DISEÑO DE UN PLAN DE GESTIÓN DE PRUEBAS PARA LOS PROYECTOS DE SOFTWARE BASADO EN LAS BUENAS PRACTICAS DE GERENCIA DE PROYECTOS.** "**,** presentado por Tania J. Andrade G, titular de la cédula de identidad N° 11.619.150, como parte de los requisitos para optar al Título de **Especialista en Gerencia de Proyectos**.

A partir de dicha revisión, considero que el mencionado Trabajo Especial de Grado reúne los requisitos y méritos suficientes para ser sometido a evaluación por el distinguido Jurado que tenga(n) a bien designar.

Atentamente,

Msc. Guillén Guédez Ana Julia.

C.I. Nº 7.599.767

_____

DEDICATORIA

A Dios por ofrecerme las oportunidades en mi vida y haberme enseñado a aprovecharlas, permitiendo alcanzar mis metas a través de algunos sacrificios, esfuerzo, dedicación y perseverancia.

A la memoria de mi hermano José Andrade, gracias por existir en mi vida. TE AMO.

A mis padres por darme su amor, paciencia, entusiasmo, apoyo y, por sobre todo, valor para seguir adelante y cumplir un sueño más en mi vida.

A mi hijo, Josué Alejandro por todo el amor sincero que me brindas, TE AMO eres fuente de mi inspiración que impulsas mi vida, para seguir luchando por un futuro mejor.

A mi esposo, Oscar Antonio que está a mi lado todo el tiempo que lo necesito apoyándome, parte de este logro te lo debo a ti. TE AMO.

AGRADECIMIENTOS

A dios por darme fortaleza y los medios para lograr la culminación de este trabajo.

A mi esposo Oscar Antonio, por su gran amor y apoyo incondicional en la realización de esta meta.

A mi hijo Josué Alejandro por su amor, cariño e inspiración en el desarrollo y logro de mis metas.

Un agradecimiento especial a la profesora Ana Julia Guillen por haber dedicado parte de su valioso tiempo en asesorarme, orientarme, guiarme en la elaboración de este Trabajo Especial de Grado.

A todos los profesores de la Dirección de Gerencia de Proyectos, por la formación que me brindaron en el área.

Gracias a todos los que colaboraron,

Tania Andrade

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
VICERRECTORADO ACADÉMICO
ESTUDIOS DE POSTGRADO
ÁREA DE CIENCIAS ADMINISTRATIVAS Y DE GESTIÓN
POSTGRADO EN GERENCIA DE PROYECTOS

DISEÑO DE UN PLAN DE GESTIÓN DE PRUEBAS PARA LOS PROYECTOS DE SOFTWARE BASADO EN LAS BUENAS PRACTICAS DE GERENCIA DE PROYECTOS.

Autor: Andrade Gil, Tania Josefina
Asesor: Guillen Guédez, Ana Julia
Año: 2016

RESUMEN

La calidad en sentido general, tanto de software como de otros tipos de productos, es un elemento que cada día se tiene más en cuenta a nivel mundial y su logro se relaciona directamente con el proceso que se emplee para obtenerla. El presente trabajo de investigación tiene por objetivo “Diseñar un plan de Gestión de Pruebas para los Proyectos de Software basado en las buenas prácticas de Gerencia de Proyectos”, con la finalidad de brindar las mejores prácticas para ejecutar las pruebas a los desarrollos del Software. Específicamente las pruebas de software permiten evaluar las soluciones y determinar el nivel de calidad que poseen. Desde el punto de vista metodológico se ubica en el tipo de investigación aplicada, ya que su finalidad es proponer un plan base para el futuro desarrollo de las pruebas de las aplicaciones de software. El resultado final de esta investigación será un documento que contiene el plan de gestión de pruebas para los proyectos de software basado en las buenas prácticas de gerencia de proyectos.

Palabras Clave: Calidad de Software, Pruebas de Software, Plan de Gestión**,** Ingeniería del Software

Línea de Trabajo: Definición y Desarrollo de Proyectos

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

En los últimos tiempos, la humanidad ha venido presenciando los grandes adelantos tecnológicos, que muchas empresas están implementando desde finales del siglo XX, y lo que va del siglo XXI. Estos cambios han estado revolucionando todos los estratos de la sociedad, la economía, la industria, entre otros; los cuales no han escapado en ningún momento a las limitaciones que pueden darse al momento de implementar estos avances tecnológicos en sus estructuras internas, debido a que las empresas no contemplaron este crecimiento tecnológico tan vertiginoso, el cual es requerido para ajustar sus negociaciones a los cambios estratégicos del negocio que se presentan en todos los procesos de la organización.

Los procesos se encuentran apoyados en el desarrollo tecnológico, influenciando desde la vida cotidiana de cualquier individuo hasta el mundo empresarial, en donde la inversión tecnológica se traduce en mayor competitividad.

En la actualidad, las empresas requieren del desarrollo y funcionamiento de un sistema de información que se ajuste a sus necesidades en la operación de sus actividades comerciales; por ello la importancia del desarrollo continuo de procesos que le faciliten en tener la herramienta automatizada que cumpla con sus expectativas de trabajo, lo cual le permitirá obtener el control oportuno de sus operaciones y un estado confiable de su situación económica.

Los sistemas de software se han convertido en algo intrínseco y natural de la vida moderna. Las personas, cada vez, usan más sistemas de software en el día a día, y en el mundo de la industria cada vez se invierte más dinero al desarrollo de software. El tamaño de los productos software ya no se mide en miles de líneas de código, sino en millones de éstas, y así, los sistemas de software se han convertido en sistemas complejos que realizan todo tipo de tareas.

En un entorno de continuos cambios y adiciones al software a través de una compilación de aplicaciones donde los requisitos deben evolucionar, las pruebas de software adquieren una naturaleza iterativa. Cada desarrollo va acompañado de un número considerable de nuevas pruebas, así como la reanudación de scripts existentes. Dado esos continuos cambios y adiciones a las aplicaciones la automatización de las pruebas de software se convierte en un importante mecanismo de control para asegurar la precisión y la estabilidad del producto a través del ciclo de vida.

El presente trabajo de investigación se desarrolló para dar respuesta a la necesidad de gestionar un Plan de Pruebas para los Proyectos de Software basado en las buenas prácticas de Gerencia de Proyectos.

El documento consta de cuatro capítulos, los cuales se describen a continuación:

El **Capítulo I El Problema**, describe el planteamiento del problema, los objetivos, la justificación del estudio, el alcance y las limitaciones de la investigación.
En el **Capítulo II Marco Teórico**, se presentan los antecedentes de la investigación, los diferentes conceptos que conforman el basamento teórico de la investigación, así como, las bases legales que sustentaron el estudio.

En el **Capítulo III Marco Metodológico**, se definen la metodología empleada, el tipo y diseño de la investigación, la unidad de análisis y las técnicas e instrumentos para la recolección de datos, así como la operacionalización de los objetivos, estructura desagregada de trabajo y los aspectos éticos del estudio.

El **Capítulo IV Ventana de Mercado**, presenta el mercado potencial de las industrias dedicadas al desarrollo del software en Venezuela, consumidores actuales o potenciales, competidores, Segmentos y Estrategias de Penetración de Competidores, agentes reguladores.

El **Capítulo V Desarrollo de los Objetivos Específicos**, contiene el desarrollo de cada uno de los objetivos específicos que persigue el presente Trabajo Especial de Grado

El **Capítulo VI Análisis de los Resultados**, se presenta un análisis de los resultados obtenidos en el desarrollo de los objetivos específicos.

El **Capítulo VII Lecciones Aprendidas**, consta de una serie de lecciones aprendidas obtenidas durante la elaboración del presente Trabajo Especial de Grado.

El **Capítulo VIII Conclusiones y Recomendaciones**, se detallan las conclusiones y recomendaciones que se llegaron luego de haber desarrollado cada uno de los objetivos específicos

Finalmente las referencias bibliográficas consultadas durante el desarrollo del Trabajo Especial de Grado.

CAPITULO I: EL PROBLEMA

De acuerdo a Arias (2012): "… Independientemente de su naturaleza, un problema es todo aquello que amerita ser resuelto. Si no hay necesidad de encontrar una solución, entonces no existe tal problema…" (p.37).

Continúa Arias (2012) detallando lo siguiente:

> "… Un problema de investigación es una pregunta o interrogante sobre algo que no se sabe o que se desconoce, y cuya solución es la respuesta o el nuevo conocimiento obtenido mediante el proceso investigativo, es entonces donde se requiere de un planteamiento que describa de manera amplia la situación objeto de estudio, ubicándola en un contexto que permita comprender su origen, relaciones e incógnitas; es por ello que cuando se formula un problema se plantea una pregunta concreta y precisa en cuanto a espacio, tiempo y población…" (pag.41 Ob. Cit.).

Considerando lo anterior, resulta de importancia capital el correcto planteamiento del problema y su correspondiente formulación por parte del investigador para dar con la solución requerida a la temática planteada.

Este capítulo describe de forma estructurada el planteamiento del problema que da origen a este trabajo, su objetivo general y objetivos específicos, la justificación, el alcance y las limitaciones de esta investigación.

1.1 Planteamiento del Problema

La evolución tecnológica ha permitido automatizar procesos que apoyan asuntos operativos y de negocios con la finalidad de aumentar la productividad y competitividad y eficiencia en el sector empresarial, razón por la cual muchas empresas deciden invertir en esta área.

La calidad del software debe tenerse en cuenta durante todo el proceso de producción del mismo, por tanto deben existir actividades que velen por su cumplimiento durante todo el ciclo de vida. Es una no conformidad esperar a que el software esté terminado para entonces aspirar a obtener la calidad en el mismo.

Uno de los principios de Humphrey (2005 y 2008), dicta que “la calidad de un producto está determinada por la calidad del proceso que se emplea para producirlo” (pp. 7-9). Algunos de los autores más reconocidos en el área de la ingeniería de software como: Beth (2009), Piattini (2009) y Pressman (2010), y especialmente, en la calidad de software, han establecido lo que sería el término propiamente dicho. Según Pressman, calidad no es más que el cumplimiento de los requisitos de funcionalidad y de desempeño establecidos explícitamente; normas de desarrollo documentadas explícitamente y características implícitas que son esperadas en todos los programas desarrollados profesionalmente (pp. 5, 8, 9) Por su parte Piattini, establece que es el conjunto de propiedades o características de un producto o servicio que le confieren aptitud para satisfacer unas necesidades expresadas o implícitas (p. 8). La gestión de dicha calidad, según apunta, debe incluir varios aspectos fundamentales como la planificación, el aseguramiento y el control de la calidad. En sentido general estos aspectos plantean la identificación de las normativas y estándares relacionados con el proyecto específico al que se le esté gestionando la calidad y a partir de ellas establecer las actividades que será necesario realizar, además de ejecutar las tareas de control, garantizando entregas de productos con cierto nivel de calidad (p.10).

El Instituto de Ingeniería Eléctrica y Electrónica IEEE (2004), las pruebas de software “consisten en verificar el comportamiento de un programa dinámicamente a través de un grupo finito de casos de prueba, debidamente seleccionados del, típicamente, ámbito de ejecuciones infinito, en relación al comportamiento esperado” (p.11). Contrario a lo que se pueda pensar una prueba exitosa es aquella en la que se consigue que el sistema falle y por tanto, se encuentran errores, aunque estos no sean todos los que posee el producto. De igual forma se puede considerar una prueba exitosa cuando es posible asegurar que no existen más errores en el sistema, cuestión que es mucho más difícil de garantizar. Este es un aspecto del que advierte muy a menudo la teoría de pruebas, ya que no es confiable asumir que se han superado todas las pruebas posibles que pueden efectuarse a un software.

Un aspecto crucial en el control de calidad del desarrollo de software son las pruebas y, dentro de estas, las pruebas funcionales, en las cuales se hace una verificación dinámica del comportamiento de un sistema, basada en la observación de un conjunto seleccionado de ejecuciones controladas o casos de prueba.

Las pruebas funcionales son aquellas que se aplican al producto final, y permiten detectar en qué puntos el producto no cumple sus especificaciones, es decir, comprobar su funcionalidad. Para realizarlas se debe hacer una planificación que consiste en definir los aspectos a examinar y la forma de verificar su correcto funcionamiento, punto en el cual adquieren sentido los casos de prueba.

Sin embargo, a pesar de la importancia de las pruebas, en muchas organizaciones no se les presta la suficiente atención o son, simplemente, omitidas en la planificación del desarrollo de productos software. Muchas veces, la presión por tener un producto terminado en una fecha establecida provoca que haya actividades, consideradas erróneamente como prescindibles, que son eliminadas o reducidas en la planificación. De esta forma, se persigue el objetivo de reducir costes y poder disponer de un producto entregable a tiempo, aunque los inconvenientes suelen ser mucho mayores que las ventajas de entregar simplemente “algo”. Así, eliminar las actividades de pruebas, en vez de acelerar el desarrollo de software, provoca que los productos software nunca puedan considerarse como realmente terminados, puesto que es posible que aparezcan continuamente defectos que deban ser resueltos una vez que el software haya sido desplegado. En este contexto, un defecto puede definirse como un comportamiento no esperado del sistema, el cual no se corresponde con los requisitos y especificaciones del mismo. De esta forma, además de afectar directamente a la satisfacción de los clientes, el costo de solucionar estos defectos aumentará a medida que se avancen en las etapas del ciclo de vida del producto software. Esto se debe a que el defecto se ha ido propagando a lo largo de las diferentes etapas, y así, un defecto que podría ser corregido por un programador en un período corto de tiempo, puede requerir la interacción demás personas (programadores, probadores de software, el cliente, los usuarios, etc.) para poder

informar, reproducir, solucionar y comprobar finalmente que el defecto ha sido corregido.

Esta situación se agrava en pequeñas y medianas empresas (PYMES) donde, habitualmente, no se dispone de recursos debidamente calificados, certificados y específicamente dedicados a la realización de las actividades relacionadas con el proceso de pruebas de software, y por ello que actualmente la organización no escapa a esta situación, debido a que se han presentado inconvenientes con los diferentes clientes, ya que cuando se realiza la instalación o las actualizaciones al sistema el mismo presenta fallas, anomalías, produciendo en el cliente insatisfacción, un colapso en las actividades económicas que el mismo realiza, de acuerdo al mercado en el que se encuentra posicionado. Es por ello que a raíz del problema planteado surgió la necesidad de Diseñar un plan de Gestión de Pruebas para los Proyectos de Software basado en las buenas prácticas de Gerencia de Proyectos.

1.1.1 Formulación del Problema

Considerando el planteamiento del problema de este trabajo de investigación, se presenta la pregunta a la situación descrita con la siguiente formulación del problema:

¿Cómo debe estar estructurado el plan de gestión de pruebas para los proyectos de Software?

1.1.2 Sistemización del Problema

Para dar respuesta a la formulación antes descrita, queda planteada una situación que conduce a formular las siguientes interrogantes:

- ¿Cuáles son las características de los proyectos de software de la Empresa objeto de estudio?
- ¿Cuáles son las mejores prácticas para la evaluación de proyectos de software?

- ¿Cómo debe estar estructurada una propuesta con las mejores prácticas de clase mundial?
- ¿Cómo debe estar conformado el plan de gestión de pruebas del área en estudio?

1.2. Objetivos de la Investigación

El propósito de esta investigación es diseñar un plan de gestión de pruebas para los proyectos de software que integre el conjunto de procesos requeridos para mejorar la eficiencia del desarrollo de las actividades de validación de productos de software, fácilmente implantables y aplicables, y que permita a la organizaciones afrontar iniciativas de mejora del proceso de pruebas software, con el objetivo de reducir el impacto de los errores

A continuación se presentan los objetivos del presente trabajo de investigación que dará respuesta al problema planteado

1.2.1 Objetivo General

Diseñar un plan de Gestión de Pruebas para los Proyectos de Software basado en las buenas prácticas de Gerencia de Proyectos.

1.2.2 Objetivos Específicos

- Caracterizar los proyectos de Software de la empresa en estudio.
- Analizar las mejores prácticas en el área de gestión de pruebas para proyectos de Software.
- Formular una propuesta basada en las mejores prácticas de Gerencia de proyectos de software.
- Elaborar las fases del plan de gestión de pruebas para la empresa objeto de estudio.

1.3. Justificación de la Investigación

La incorporación y el uso de las nuevas tecnologías son, sin lugar a dudas, factores clave para lograr la competitividad del país, el desarrollo del sector empresarial y mejoras en la educación, la salud y la calidad de vida de los ciudadanos.

Debido a los problemas generados cuando se realizan actualizaciones del sistema o instalación del sistema en los diferentes clientes, es necesario diseñar un plan de gestión para los proyectos de software que permita evitar estos inconvenientes generados garantizándole su crecimiento competitivo, y el avance en sus negociaciones, contribuyendo a que sus tomas de decisiones se basen en los resultados que se tienen en el trabajo del día a día.

La investigación que se presentó a través de este estudio, aporto un plan de gestión que mejoro los procesos de ejecución de pruebas de la empresa en estudio, y obtuvo las siguientes ventajas:

- Detección oportuna de las debilidades presentadas en los proyectos.
- Garantizar un mecanismo de mejora continua en la organización, que permita auditar desarrollos de software internos, planificar la estrategia de ingeniería del software de la organización, etc.
- Asegurar la precisión y la estabilidad del producto.
- Asegurar primeramente que se ha cumplido con los requisitos de funcionalidad previstos para la implantación e implementación del sistema.
- Reducción en el tiempo de entrega y el incremento en la eficiencia de pruebas

Así mismo, al identificar el grado de satisfacción del cliente con respecto al sistema ofrecido, resulta fundamental a la hora de realizar un balance sobre el desempeño de la empresa. En tal sentido, la medida de satisfacción representa un mecanismo de retroalimentación para la organización, que le permitio tomar medidas adecuadas para el desarrollo futuro en la captación de los clientes, todo

esto en función de las estrategias que deben llevarse a cabo por la gerencia de la organización.

El plan de gestión que se propuso aporto la optimización de los procesos actuales, conjugando esfuerzos orientados hacia la incorporación de una tecnología que permita proveer, facilitar la definición y cumplimiento de los objetivos de calidad del producto y fortalecer la relación de la empresa con sus clientes, beneficiando a estos, para seguir el funcionamiento de sus actividades económicas evitando un posible colapso de las mismas, brindándoles un acceso rápido y seguro de la información, permitiendo así el incremento de la satisfacción del cliente frente al producto entregado.

1.4. Alcance y Delimitaciones de la Investigación

El alcance de la presente investigación incluyo los procesos para lograr el Diseño de un plan de Gestión de Pruebas para los Proyectos de Software en las buenas prácticas de Gerencia de Proyectos.

Por otro lado el desarrollo de este proyecto representa una propuesta a la Gerencia de Tecnología de la Empresa en estudio y abarca hasta el diseño del plan de gestión de pruebas, queda de parte de la organización su implantación o uso del diseño del plan de gestión, así como su éxito dependerá de los lideres de proyectos y de cómo aplicaran los lineamientos aquí planteados.

CAPÍTULO II: MARCO TEORICO

El marco teórico es la base que sustento la investigación, los cimientos basados en revisiones bibliográficas, que permitió elaborar toda la teoría que fue la clave para poder desarrollar los objetivos planteados.

En este capítulo se presentaron los antecedentes, fundamentos teóricos, bases legales, definición de términos referidos al tema de pruebas de software de productos.

A continuación se resumen una serie de trabajos de grado relacionados con el área de investigación, que fueron consultados y usados como soporte para dar respuesta al problema planteado en el Capítulo I: Planteamiento del Problema.

2.1 Antecedentes

Los antecedentes de la investigación representan estudios previos que están relacionados con el problema planteado. Según Arias (2012), los antecedentes reflejan los avances y el estado actual del conocimiento en un área determinada y sirven de modelo o ejemplo para futuras investigaciones (p. 106).
A continuación se muestra la descripción de Trabajos Especiales de Grado, del Postgrado de Gerencia de Proyectos de la Universidad Católica Andrés Bello, relacionados con el Área de Gestión de pruebas, que sirvieron como antecedentes de la investigación. La descripción recoge los aspectos más importantes de cada documento y sus respectivos comentarios según sea el caso.

La Tesis de Fernández (2015), titulado "**Aplicación de técnicas de pruebas automáticas basadas en propiedades a los diferentes niveles de prueba del software**", para optar al Título de Doctor en Computación de la Universidad de Coruña, el cual tuvo como objetivo general: Definir un conjunto de metodologías y técnicas de pruebas que puedan ser usadas a lo largo del ciclo de vida de un producto software. Estas metodologías y técnicas de pruebas estarán adaptadas

según la etapa del ciclo de vida en la que se encuentra el producto software (implementación de componentes unitarios, integración de varios componentes, etc.), pero siempre tendrán como objetivo principal conseguir que las pruebas sean más eficientes, es decir, invertir menos tiempo en la realización de las mismas, y más efectivas, esto es, encontrar más defectos lo más pronto posible en el desarrollo. De esta forma, la idea principal es que, usando este tipo de técnicas, se aumente la calidad de los productos software.

De esta forma, esta investigación ofrece una serie de metodologías de pruebas, integradas en el proceso de desarrollo, las cuales se basan en aproximaciones de pruebas automáticas basadas en propiedades. Además, se han proporcionado las herramientas necesarias para llevar a cabo estas metodologías, las cuales permiten llevar a la práctica los conceptos explicados.

Palabras clave: Desarrollo de Software, implementación de componentes, metodologías de pruebas, Pruebas de Software.

La Tesis de Gutierrez (2011), titulado "**Generación de Pruebas del Sistema a Partir de la Especificación Funcional**", para optar al Título de Doctor por la Universidad de Sevilla, el cual tuvo como objetivos específicos: a) Estudiar la formalización de los artefactos utilizados en el proceso de generación de pruebas del sistema y, en concreto, de los requisitos funcionales y de los casos de prueba funcionales; b) Ampliar el uso de estándares que faciliten la interacción del proceso de generación de pruebas en un proceso de ingeniería guiada por modelos y la aplicación de herramientas existentes; c) Especificar un proceso de generación de pruebas funcionales del sistema que utilice artefactos formalizados; d) Definir la automatización de la generación de pruebas funcionales del sistema de manera semiautomática o automática; e) Validar el proceso definido mediante la aplicación a un ejemplo concreto.

El aporte de esta investigación es la presentación de dos perfiles de UML que permiten representar la información y semántica definida en este trabajo de tesis

para los requisitos funcionales y las pruebas funcionales del sistema como una extensión del propio UML. Se han traslado a dichos perfiles los mismos conceptos definidos en los metamodelos anteriores. Para ello, ha sido necesario estudiar la jerarquía y los elementos de UML para su extensión, lo que permite trabajar con cualquier herramienta que recoja dicho mecanismo de extensión.

Palabras clave: Pruebas Funcionales, Casos de Prueba, Ingeniería de Software, Calidad del Software.

La Tesis de Segura (2010), titulado "**Functional And Perfomance Testing of Feature Model Analysis Tools (Pruebas funcionales y Modelo de Análisis de Herramientas)** ", para optar al Título de Doctor por la Universidad de Sevilla, el objetivo de esta tesis es combinar el conocimiento de las comunidades de pruebas software y modelado de características en una serie de contribuciones que permitan llevar el análisis automático de modelos de características a un nuevo nivel de madurez. Con este fin, en esta memoria se presentan una serie de técnicas, algoritmos y herramientas para la realización de pruebas funcionales y de rendimiento en herramientas de análisis de modelos de características. Estas contribuciones son el resultado de la aplicación de varias técnicas clásicas e innovadoras de pruebas software en el contexto de análisis de modelos de características. Para facilitar las pruebas funcionales, se presentan un conjunto de casos de prueba así como un generador automático de datos de prueba para la detección de errores en las herramientas de análisis. En lo que se refiere a pruebas de rendimiento, presentamos un algoritmo evolutivo para la generación de modelos de características difíciles de procesar en términos de tiempo y memoria requeridos para su análisis. Etas herramientas han sido evaluadas experimentalmente de forma rigurosa demostrando la viabilidad de la propuesta.

Palabras clave: Pruebas Funcionales, Ingeniería de Software, Pruebas de Software, Calidad de Software

En el mismo orden se consultaron varios artículos de revistas, entre los cuales se seleccionaron los siguientes:

El primer artículo citado fue publicado por Palacio (2009) en el área de Validación del Software, titulado “Generating Functional Testing Case Method in Software Development (Método para Generar casos de Prueba Funcional en el Desarrollo de Software) “. Mediante esta publicación el autor presenta un método propuesto para derivar casos de prueba funcional, a partir de un ensamble entre las aproximaciones estudiadas y la experiencia de la empresa con la cual se desarrolla este proyecto cofinanciado.

La propuesta objeto de este artículo recoge los métodos existentes y la experiencia empresarial para generar un método con un nivel de formalidad adecuado, y reduce algunas desventajas como el nivel de dificultad y la cantidad de modelos intermedios, evidenciados en las otras aproximaciones.

Palabras clave: Pruebas de software, Casos de prueba, Ingeniería de Software, Pruebas Funcionales

El segundo y último artículo consultado como antecedente es publicado por **Jústiz, Gómez & Delgado (2013)** en el área de Validación del Software, titulado “Testing process for software products at a quality laboratory (Proceso de pruebas para productos de software en un laboratorio de calidad) “. En esta publicación los autores resaltan lo siguiente:

1. La implementación de un proceso de pruebas y su puesta en marcha en un Laboratorio de Calidad, permite elevar la calidad, valga la redundancia, de los productos obtenidos durante el proceso de producción así como organizar y gestionar la actividad de las pruebas funcionales en el ciclo de vida.
2. La información que se obtiene de este proceso ahora se encuentra controlada y es uniforme en soluciones similares lo que permite un control de la calidad más efectivo.

3. Con este procedimiento establecido es posible detectar y dar seguimiento a los defectos de manera que las soluciones que se entregan a los clientes de la organización poseen un mayor nivel de calidad.
4. Los roles involucrados en el proceso tienen claro cuáles son su responsabilidades y el equipo de pruebas está integrado en su mayoría por especialistas externos al equipo de desarrollo. Con esto se logra una mayor objetividad en la comprobación de los productos ya que se disminuye la posibilidad de omitir la prueba de alguna funcionalidad, como suele suceder cuando es el propio desarrollador quien prueba lo que ha concebido. En sentido general, la definición del proceso de pruebas constituye un paso de avance en la tarea de garantizar, al menos, cierto nivel de calidad para los productos que se construyen en la organización. El alcance de esta propuesta no se limita a la organización en cuestión si no que puede aplicarse en cualquier laboratorio de pruebas, ya sea interno, en un proyecto de desarrollo, o externo, a nivel de organización o empresa. En principio está pensado para empresas de desarrollo de software ya que sus actividades comprenden la evaluación de soluciones de software, sin embargo pudiera aplicarse a otros procesos de producción si se realizan las modificaciones pertinentes en la modelación de las actividades y los artefactos

Palabras clave: Calidad de Software, Pruebas de Software, Proceso de Pruebas, Niveles de Prueba.

2.2. Fundamentos Teóricos

En esta sección se presentaron las definiciones y los temas según el criterio de varios autores, que conforman el fundamento teórico de la investigación.

Proyecto

Para Lledó y Rivarola (2007) un proyecto es “un desafío temporal que se enfrenta para crear un único producto o servicio” (p. 4); el PMI (2013) lo define como “un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado

único” (p. 3); para Chamoun (2002) un proyecto es “un conjunto de esfuerzos temporales, dirigidos a generar un producto o servicio único” (p. 27). Al revisar otros autores se puede notar que estos son concordantes con las definiciones antes citadas, la gran mayoría de ellos concuerda en que un proyecto es un esfuerzo llevado a cabo en un período de tiempo finito con el objeto de generar un producto o servicio de carácter único. El proyecto llega a su fin cuando el objetivo de éste es alcanzado, en el tiempo que para ello se ha definido. El término temporal no necesariamente quiere decir corto plazo, el período de tiempo podrá tener la duración que sea necesaria. El resultado final de un proyecto podrá ser de cualquier tipo, desde una obra civil como un puente, hospital, autopista, hasta la producción de un comercial para la prensa, televisión etc.

La Gerencia de Proyectos

Para el PMI (2013), la gerencia de proyectos o dirección de proyectos se define como “la aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para cumplir con los requisitos del mismo” (p. 5), Chamoun (2002) coincide casi totalmente con la definición anterior, siendo para este “la aplicación de conocimientos, habilidades, técnicas y herramientas a las actividades de un proyecto, con el fin de satisfacer, cumplir y superar las necesidades y expectativas de los involucrados” (p. 39). La administración o gestión de proyectos basada en el modelo expuesto en el PMI (2013), se lleva a cabo mediante la realización de un conjunto de procesos que cubren diez áreas de conocimiento a través del ciclo de vida del proyecto, permitiendo de esta forma que este avance de manera eficaz hacia los objetivos planteados; las diez áreas de conocimiento son: integración, alcance, tiempo, costo, calidad, recursos humanos, comunicación, riesgo, adquisiciones e interesados.

El área de integración considera dentro de las herramientas que deben útil izarse los sistemas de información para la dirección de proyectos, y como parte fundamental de un sistema de información hoy en día, se debe mencionar las

herramientas informáticas (software), de aquí la importancia de un plan para el diseño de una solución de este tipo.

Los procesos de la Gerencia de Proyectos

Según el PMI (2013), un proceso es un conjunto de acciones y actividades interrelacionadas realizadas para obtener un producto, resultado o servicio predefinido. Eso indica que todo proceso influye una sucesión de actividades que, necesariamente tienen cada una de ellas alguna actividad precedente, y/o alguna actividad siguiente. Cada proceso se caracteriza por sus entradas, por las herramientas y técnicas que puedan aplicarse y por las salidas que se obtienen.
Los procesos de dirección o gestión de proyectos se agrupan en cinco categorías como grupo de procesos de la Dirección de Proyectos:

- ✓ **Procesos de Iniciación:** relacionado con un nuevo proyecto o una nueva fase de un proyecto existente.
- ✓ **Procesos de Planificación:** procesos requeridos para establecer el alcance del proyecto, refinar y orientar las acciones necesarias para alcanzar los objetivos y obtener el logro del proyecto.
- ✓ **Procesos de Ejecución:** procesos que se requieren para la coordinación de los recursos materiales, equipos y humanos para completar los trabajos definidos en el plan de gestión de proyecto.
- ✓ **Procesos de Seguimiento y Control:** procesos requeridos para dar seguimiento, analizar, medir y regular el progreso y el desempeño del proyecto. Identificar áreas que requieran cambios y la toma oportuna de acciones correctivas cuando sea necesario.
- ✓ **Procesos de Cierre:** procesos para formalizar y finalizar todas las actividades para la aceptación de producto, servicio o resultado a fin de cerrar formalmente el proyecto o una fase del mismo.

Tipos de Metodologías de Gerencia de Proyectos

Una buena práctica es dividir el proyecto en varias fases, la suma de las fases es el ciclo de vida de un proyecto. Estas fases varían de una metodología a otra, pero en general incluyen el inicio, planificación, implementación, monitoreo y cierre. Un proyecto tiene que completar exitosamente cada fase antes de iniciar la siguiente, esto hace que el ciclo del proyecto tenga mejor control y construya los nexos apropiados con el entorno interno y externo.

Las metodologías constituyen un marco de referencia que registran las recomendaciones sobre las mejores prácticas para ejecutar los proyectos, entre las metodologías consideradas se encuentran las siguientes:

Metodología Front-End-Loading (FEL)

Es una metodología para proyectos de inversión que consiste en un conjunto de procesos para el desarrollo de proyectos competitivos basados en la consideración gradual y comprensiva de todos los factores clave que permitan traducir la estrategia de una compañía en un proyecto clave. Esta metodología permite reducir costos y mantener el proyecto dentro de los tiempos de ejecución planificados.

El término front-end-loading, fue acuñado por la compañía DuPont en 1987, y usado por las industrias químicas, refinerías y gas. A partir de un trabajo de benchmarking desde 1993 hasta 2003, y sobre la base de la experiencia en varias empresas consultadas que usaban la definición y desarrollo para sus proyectos, la Independent Project Analisys Inc. (IPA), empresa de ingeniería y consultoría en gerencia de proyectos, identificó las fases de una metodología a la que denominó ciclo FEL (Front End Loading), a otro grupo de fases para la implatación kas denominó ciclo EPCC (Engineering, Procurement,Construction, Commissioning), y a la fase de operación como última fase. La metodología FEL fue presentada por la IPA Inc en las 30va y 32va. Conferencia anual de Ingeniería y Contratación de Construcción (Annual Engineering & Construction Contracting Conference) en los años 1998 y 2000 respectivamente.

Las fases de la metodología FEL tal como se puede ver en la figura N° 1 también son conocidas como: fase de visualización (Identificación de Oportunidades), fase de conceptualización (Selección de Alternativas) y fase de definición (Planificación del Proyecto). El producto del proceso FEL, es el paquete de las bases de diseño de requisitos particulares para soportar la ingeniería de detalle del proyecto del ciclo EPCC.

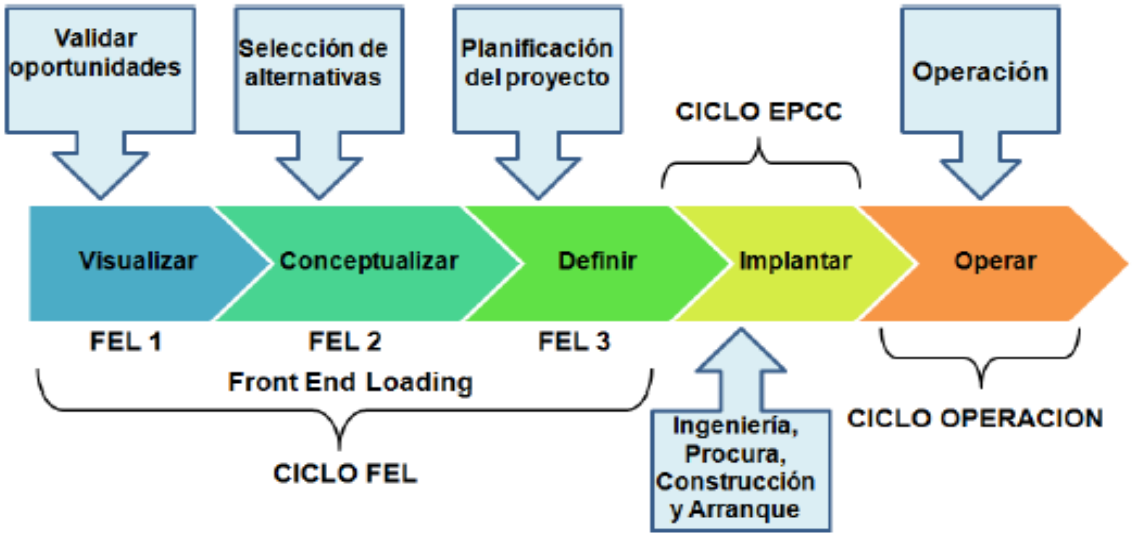


Figura N° 1. Diagrama del ciclo FEL-EPCC-Operación

Fuente: Adaptado de Guías de Gerencia para Proyectos de Inversión de Capital (1997)

La IPA Inc también define tres fases de ingeniería que denomina FEED (Front End Engineering Developement), como: Ingenieria conceptual (fase de conceptualización), Ingeniería básica (fase de definición), Ingeniería de detalle (fase de ingeniería). Solo las fases de ingeniería conceptual y la de ingeniería básica, están presentes en el ciclo FEL; (FEL 2, FEL 3 respectivamente), la fase de ingeniería de detalle pertenece al ciclo EPCC.

Un plan de proyecto FEL se crea en tres fases distintas (FEL 1, FEL 2, FEL 3), para asegurar la inversión y unos análisis cuidadosos al proyecto. Durante las primeras dos fases (FEL 1 y FEL 2), “Visualización y Conceptualización” se examinan todas las oportunidades posibles del negocio, se exploran los beneficios

y los riesgos de cada oportunidad, y se refina el alcance del proyecto. Durante la tercera fase (FEL 3), “Definición”, se ejecuta la ingeniería básica para la mejor opción.

Como se puede observar en la figura 1, cada fase de la metodología tiene una denominación que la distingue de las otras. Visualización, conceptualización o definición; y un propósito general muy bien definido para la toma de decisiones estratégicas que identifican valor.

Fases de la Metodología FEL

Fase FEL I – Fase de Visualización

En esta fase se identifican las oportunidades de negocio y se generan las opciones técnicas y económicamente factibles de las propuestas o ideas para el proyecto. Así mismos se identifican los riesgos generales y las mejores estrategias que permitan optimizar los resultados del proyecto. Se presenta un estimado de costo clase V. Al finalizar esta fase, se genera un escenario para su posterior aprobación.

Fase FEL II – Fase de Conceptualización

Una vez aprobado el DSD de la fase de visualización y los recursos necesarios, se continúa con la fase de conceptualización. En esta fase se evalúa(n) el (los) escenario(s) u opciones y se selecciona aquel que genere mayor valor. Se inicia la planificación del proyecto con la ingeniería conceptual y se evalúa y selecciona la alternativa tecnológica. Se profundiza en la identificación de los riesgos para minimizar la incertidumbre en los stakeholders. Se presenta un estimado de costo clase IV.

Fase FEL III – Fase de Definición

Una vez aprobado el DSD de la fase de conceptualización y los recursos necesarios, se continúa con la fase de definición. En esta fase, se realiza la ingeniería básica para completar el alcance de planificación y diseño de la opción seleccionada. Se profundiza en la evaluación de los riesgos para minimizar la incertidumbre en los stakeholders. Se afina el estimado de costos hasta precisar la solución estratégica de contratación e implantación de entre el -5% +15%, para asegurar que el proyecto esté bien estructurado y listo para solicitar su autorización y los recursos para su ejecución. Se elabora el plan de ejecución para la EPCC.

Procesos del Ciclo Front- End - Loading (FEL)

El proceso se inicia cuando la idea del proyecto es concebida por un resultado de los análisis del ambiente interno-externo del negocio, o del análisis de una matriz FODA (Fortalezas, Oportunidades, Debilidades y Amenazas); o unas iniciativas de un grupo de ingeniería, o de un grupo de desarrollo o de la unidad de negocio.

Es importante la interacción de los stakeholders para incorporar los cambios necesarios y ensamblar el paquete base de diseño requerido de la fase para su correspondiente aprobación.

Cada fase de procesos, antes de ser iniciada, debe estar correctamente planificada, y su fase anterior auditada y aprobada. Cada una debe cumplir una serie de actividades y puntos de verificación y control, y así obtener la correspondiente autorización de los niveles de autoridad de la organización, antes de avanzar a la siguiente fase de procesos y comprometer recursos del proyecto.

En cada una de las fases, se van incorporando elementos de información y análisis que permitan una mayor definición del alcance, una minimización de los riesgos e incertidumbre, así como un estimado de costos y programas de ejecución más precisos.

Se requiere de equipos multidisciplinarios que interactúen para desarrollar cada fase con sus respectivos entregables completamente estructurados. Estos entregables son la base fundamental del **Documento de Soporte de Decisión**

(DSD). Este documento le permite a los diferentes niveles de autoridad analizar el estatus de la fase en cuestión para su conformidad y aprobación, así como también las consideraciones técnicas de la fase respectiva. El documento de soporte de decisión representa un hito fundamental para obtener la aprobación de los recursos necesarios para avanzar hacia la fase siguiente.

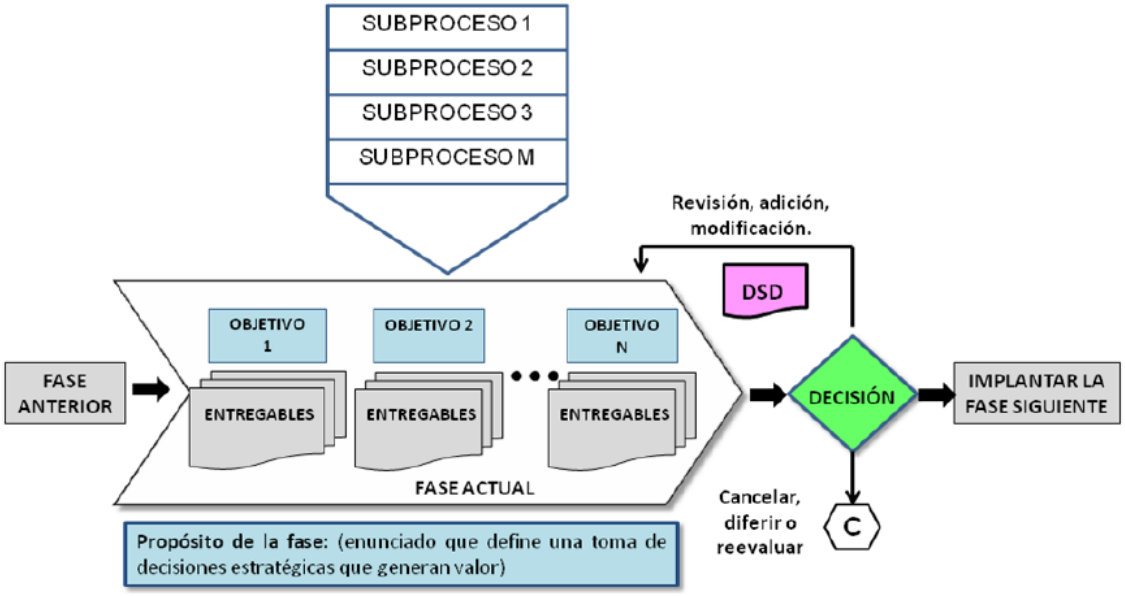


Figura N° 2. Ciclo de procesos de una fase FEL

Fuente: Adaptado de IPA INC

La figura muestra el ciclo de procesos de la fase de la metodología.

Cada fase está conformada por un conjunto de objetivos bien definidos según las características estratégicas que tenga el propósito y correctamente alineados con el proyecto y el negocio.

En la figura 2 se identifican los siguientes subprocesos de la fase:

- Fase actual: identifica el nombre dado a la fase por la IPA Inc., Visualización, Conceptualización y Definición.
- Los objetivos: identificados en la figura como objetivo 1, objetivo 2, objetivos N, que son definidos por los niveles de autoridad de la organización y alineados con la estrategia del propósito de la fase.

- Subprocesos: identificados como subproceso 1, subproceso 2, subproceso M, son un conjunto de actividades particulares para lograr objetivos específicos de la fase, se debe completar un conjunto de actividades cuyo subprocesos son definidos por la organización del proyecto.
- Entregables: que identifican la documentación como son planos, memorias descriptivas, estimados de costos, entre otros, que son el resultado de los procesos de cada fase, y que serán el soporte para el análisis y la toma de decisiones sobre la fase.
- Decisiones: en cada finalización de fase se puede tomar una de las siguientes acciones: aprobar los resultados de la fase y obtener los recursos para avanzar hacia la siguiente fase; o ejecutar el proyecto según sea el caso; cancelar o diferir el proyecto.
- DSD: que identifica *el Documento de Soporte de Decisión* para conformidad y aprobación de la fase para pasar a la siguiente fase, identificado en la figura con la letra C dentro de un hexágono; o devolver la documentación de la fase al equipo de trabajo para su revisión, modificación o para completar o añadir las observaciones, consideraciones y/o elementos de las opciones evaluadas.

  El DSD se podrá conformar de varias formas, según sea el tipo de proyecto que se esté ejecutando. Su contenido es diferente para cada fase.

Índice del Grado de Definición de Proyectos (PDRI)

Según Hackney (1992), el PDRI es una herramienta gerencial que provee un indicador sobre el grado de definición obtenido en el alcance de un proyecto. Esta técnica originalmente fue desarrollada en su forma muy básica por Hackney (1992); la cual comprendía la categorización de los ítems más importantes de un proyecto y presentados en detalle mediante una lista de chequeo (cheklist) para la planificación del proyecto.

Actualmente existen tres conceptos del PDRI, dos definidos por la CII, y uno adaptado definido por la Oficina de Gerencia de Proyectos Ambientales (EM-6),

del Departamento de Energía de USA. A continuación se describirán los fundamentos teóricos de los PDRI más importantes.

1. PDRI de la CII

Según Hackney (1992), es una herramienta a utilizar para evaluar el grado de completación del alcance de un proyecto, en cualquier punto de las fases anteriores a la autorización para realizar la ingeniería de detalle, procura y construcción del proyecto.

En el año 1994 el Instituto de la Industria de la Construcción (Construction Industry Institute, CII en inglés), constituye un equipo de investigación formado por ingenieros e investigadores de la CII y de la Universidad de Austin en Texas, USA, para definir un estándar para la planificación de anteproyectos, de tal forma que, pudiera alcanzar mejor los objetivos del negocio y del proyecto. La CII presenta dos versiones de PDRI, uno en el año 1996 para proyectos industriales, y otro en el año 1999, para proyectos de la industria de la construcción, en respuesta a las necesidades de los dos sectores, Hackney (1992).

Algunos beneficios del PDRI

- Lista de chequeo que puede usar el equipo para determinar los pasos necesario a seguir en la definición del alcance del proyecto.
- Lista de terminología estandarizada en la industria de la construcción.
- Estándar industrial para apreciar la completitud del paquete de definición del alcance del proyecto y para facilitar la evaluación del riesgo y predecir la escalación y las posibles disputas.
- Un medio para monitorear el progreso en varias etapas durante la planificación pre-proyecto.
- Una herramienta que ayuda a la comunicación entre propietarios y contratistas de diseño resaltando las áreas pobremente definidas en el paquete de definición del proyecto.
- Un medio para los participantes en el equipo del proyecto para reconciliar las diferencias usando una base común de evaluación.

- Una herramienta de trabajo para las empresas y los individuos.
- Una herramienta de benchmarking para las empresas para ser usada en la evaluación de la completitud de la definición del alcance versus el rendimiento de proyectos pasados, ambos en la compañía o externos, con el objeto de predecir la probabilidad de éxito en futuros proyectos, Hackney (1992).

El PDRI para proyectos industriales consiste en 70 elementos ponderados, divididos en tres secciones principales y 15 categorías.

El PDRI para proyectos de la industria de la construcción, es una matriz compuesta de 64 elementos, agrupados en 11 categorías, y éstas categorías son agrupadas en tres secciones principales.

Según las experiencias de la CII en una muestra de 62 proyectos industriales, su análisis ha revelado una diferencia significativa de desempeño muy bajo entre proyectos cuyo puntaje ha estado por encima de 200; y un desempeño muy alto, entre proyectos exitosos cuyo puntaje ha estado por debajo de 200.

Para propósitos del PDRI, proyectos industriales abarcan, pero no se limitan, a los siguientes tipos de facilidades:

- Facilidades de producción de crudo y gas.
- Refinerías
- Plantas químicas
- Plantas Farmacéuticas
- Industria del hierro y aluminio
- Plantas de energía
- Plantas de manufactura
- Plantas procesadoras de Alimentos
- Plantas Textiles

2. EM- PDRI de la DOE

En febrero del año 2001, la Oficina de Proyectos Ambientales (EM-6), del Departamento de Energía de Estados Unidos, (DOE en inglés), presentó una versión de PDRI denominada EM-PDRI, similar a la del CII, para propósitos específicos de mejorar la planificación de sus proyectos en la EM-6.

El EM-PDRI es una matriz compuesta de 77 elementos distribuidos en cinco áreas clave predeterminadas, identificadas como: costos, programación, alcance técnico, planificación y control, y factores externos. En la tabla N° 1 se muestra la distribución numérica de los elementos por áreas.

Tabla N° 1. Distribución numérica de elementos por áreas del EM-PDRI

ÁREAS	CANT. ELEM.
COSTOS	7
PROGRAMACIÓN	7
ALCANCE TÉCNICO	39
PLANIFICACIÓN Y CONTROL	19
FACTORES EXTERNOS	5
TOTAL	77

Fuente: Adaptado de Manual EM-PDRI (2001)

Cada área contiene agrupado un conjunto de elementos específicos, los cuales cada uno tiene asociado un valor de definición de madurez y un criterio de asignación cuantitativo y cualitativo, que en conjunto determinan el grado de madurez alcanzado por ese elemento.

El valor de madurez es una asignación numérica del cero al cinco; donde el cero significa trabajo no comenzado, y el valor cinco criterios completamente alcanzados. En la Tabla N° 2 se muestra la definición del criterio de valor de madurez.

Tabla N° 2. Definición del valor de madurez del EM-PDRI

VALOR DE MADUREZ	CRITERIO CUALITATIVO	CRITERIO CUANTITATIVO
N/A	No Aplica	-
0	Trabajo No comenzado	0
1	Trabajo Iniciado	1-20
2	Concepto Definido	21-50
3	Trabajo en Detalle	51-80
4	Diseño Final	81-95
5	Criterio Completamente Alcanzado	96-100

Fuente: Adaptado de Manual EM-PDRI (2001)

En la tabla N° 2, se utilizan valores predeterminados por el EM-PDRI, sin embargo el gerente o equipo de proyecto son libres de utilizar con una cierta discreción para puntuar el grado de un elemento en particular, basado sobre la documentación de soporte, experiencia y conocimiento de la descripción de ese elemento del proyecto.

Algunos elementos del proyecto se consideran que no cumplirán las expectativas de completación en algunas fases del proyecto, en este sentido, deben ser evaluados *No aplicables (N/A)*. Esta valoración de madurez tendrá un valor cero para efectos del valor actual (score actual) de ese elemento.

El puntaje global es de 1000 puntos, y este es obtenido por la suma de las combinaciones de las valoraciones puntuales de cada área en la fase final del proyecto.

En la tabla N° 3, se presenta la valoración predeterminada de cada área del EM-PDRI.

Tabla N° 3. Valoración de las áreas del EM-PDRI

ÁREAS	N° DE PUNTOS
COSTOS	150
PROGRAMACIÓN	150
ALCANCE TÉCNICO	400
PLANIFICACIÓN Y CONTROL	200
FACTORES EXTERNOS	100
TOTAL	1000

Fuente: Adaptado de Manual EM-PDRI (2001)

Para determinar el valor actual o valor esperado de cada elemento, se multiplica su factor de peso predeterminado por su valor de madurez adecuado. El total de la suma de los valores actuales o esperados de los elementos determina el valor del área correspondiente para ese momento de avance de la fase.

El EM-PDRI define para cada fase un factor de decisión critica (CD), como se muestra en la figura N° 3. Este concepto es muy apropiado como soporte del proceso de evaluación para las fases de la metodología FEL.

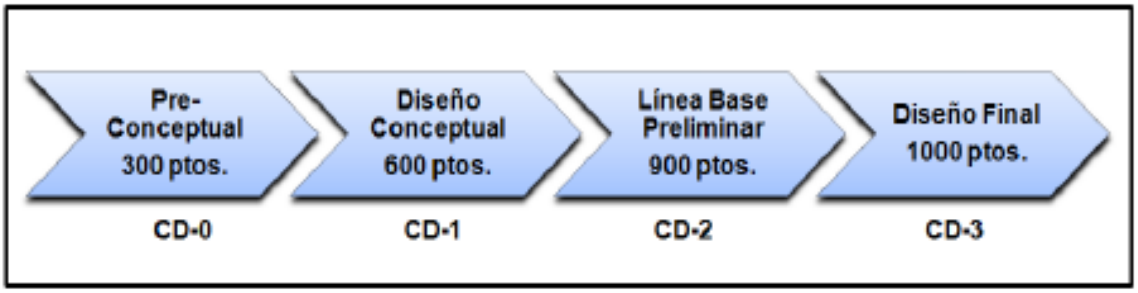


Figura N° 3. Fases del EM-PDRI para proyectos tradicionales
Fuente: Adaptado de Manual EM-PDRI (2001)

Ciclo de Vida de los Proyectos.

En un término amplio el ciclo de vida de un objeto o sistema es el conjunto de etapas, fases o estadios por el cual evoluciona este objeto o sistema, desde su nacimiento hasta su muerte. Para el PMI (2013) el ciclo de vida de los proyectos se define como “la serie de fases por las que atraviesa un proyecto desde su inicio hasta su cierre. Las fases son generalmente secuenciales y sus nombres y números son determinados por las necesidades de gestión y control de la organización u organizaciones que participan en el proyecto. Las fases son generalmente acotadas en el tiempo, con un inicio y un final o punto de control. Un ciclo de vida se puede documentar dentro de una metodología. Se puede determinar o conformar el ciclo de vida del proyecto sobre la base de los aspectos únicos de la organización, de la industria o de la tecnología empleada. Mientras que cada proyecto tiene un inicio y un final definido, los entregables específicos y las actividades que se llevan a cabo variarán ampliamente dependiendo del

proyecto. El ciclo de vida proporciona el marco de referencia básico para dirigir el proyecto, independientemente del trabajo específico involucrado" (p. 38).

Entender cuál es la finalidad de cada una de las fases del proyecto, permite saber cómo evolucionará el proceso mediante el cual se desenvolverá el proyecto y cuáles serán los entregables que deben ser producidos. De aquí se desprende que es importante diferenciar el ciclo de vida del proyecto del ciclo de vida del producto o servicio que el proyecto pretende desarrollar. El ciclo de vida del producto o servicio razón del proyecto, es el conjunto de fases mediante las cuales se desarrollará el producto o servicio, para el caso de este trabajo el ciclo de vida del desarrollo de software es el conjunto de fases mediante las cuales se lleva a cabo la construcción de la aplicación de software.

Lledó y Rivarola (2007), definen el ciclo de vida como un conjunto de fases que permiten hacer más eficiente la administración del proyecto, produciéndose distintos entregables en cada una de ellas. La figura N° 4 resume de manera concreta las cinco fases del modelo propuesto por Chamoun (2002).

.

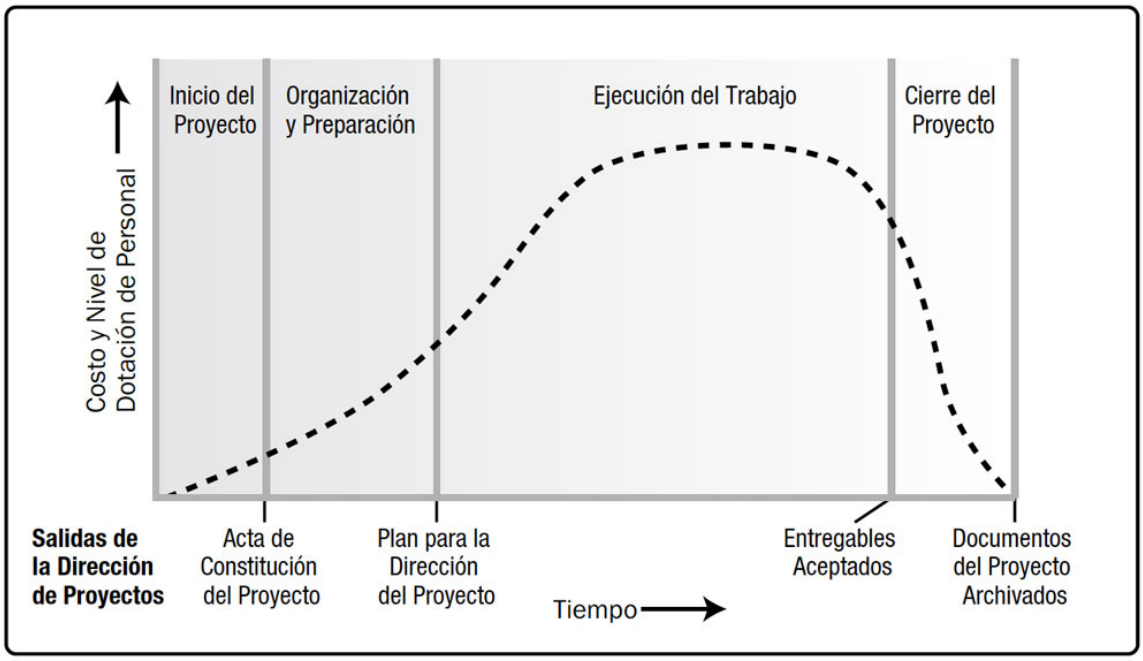


Figura N° 4. Ciclo de Vida del Proyecto

Fuente: PMI 2013

Ciclos de Vida Predictivos

Según PMI (2013) son aquellos en los cuales el alcance del proyecto, el tiempo y costo requeridos para lograr dicho alcance, se determinan lo antes posible en el ciclo de vida del proyecto. Estos proyectos atraviesan una serie de fases secuenciales o superpuestas, donde cada fase suele enfocarse en un subconjunto de actividades del proyecto y en procesos de la dirección del proyecto. El trabajo realizado en cada fase normalmente es de naturaleza diferente al realizado en las fases anteriores y subsiguientes, y por lo tanto la composición y habilidades requeridas del equipo del proyecto puede variar de una fase a otra.

Generalmente se opta por ciclos de vida predictivos cuando el producto a entregar se comprende bien, existe una base práctica significativa en la industria, o cuando un producto debe ser entregado en su totalidad para que tenga valor para los grupos de interesados.

Ciclos de Vida Iterativos e Incrementales

Para el PMI (2013) son aquellos en los cuales, dentro de las fases del proyecto (también llamadas iteraciones), se repiten de manera intencionada una o más actividades del proyecto a medida que aumenta el entendimiento del producto por parte del equipo del proyecto. Las iteraciones desarrollan el producto a través de una serie de ciclos repetidos, mientras que los incrementos van añadiendo sucesivamente funcionalidad al producto. Estos ciclos de vida desarrollan el producto de forma iterativa y con incrementos graduales.

Generalmente se opta por los ciclos de vida iterativos e incrementales cuando una organización necesita gestionar objetivos y alcances cambiantes, para reducir la complejidad de un proyecto o cuando la entrega parcial de un producto beneficia y genera valor para uno o más grupos de interesados sin afectar el entregable o conjunto de entregables finales. Los proyectos grandes y complejos se ejecutan a menudo de modo iterativo para reducir el riesgo, al permitir que el equipo incorpore retroalimentación y lecciones aprendidas entre iteraciones.

Ciclos de Vida Adaptativos

Según el PMI (2013), los ciclos de vida adaptativos (también conocidos como métodos orientados al cambio o métodos ágiles) pretenden responder a niveles altos de cambio y a la participación continua de los interesados. Los métodos adaptativos también son iterativos e incrementales, pero difieren de los anteriores en que las iteraciones son muy rápidas (normalmente con una duración de 2 a 4 semanas) y de duración y costo fijos. Los proyectos adaptativos generalmente ejecutan varios procesos en cada iteración, aunque las iteraciones iniciales pueden concentrarse más en las actividades de planificación.

Generalmente se opta por los métodos adaptativos en entornos que cambian rápidamente, cuando los requisitos y el alcance son difíciles de definir con antelación y cuando es posible definir pequeñas mejoras graduales que aportarán valor a los interesados.

Ingeniería de software

La ingeniería de software es la rama de la ingeniería que tiene como fin la creación y modificación de soluciones informáticas.

Para Pressman (2010) la ingeniería de software es “el establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales” (p. 11). Sommerville (2005) define a la ingeniería de software como “una disciplina que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación, hasta el mantenimiento de éste después que se utiliza”,

Ciclo de Vida del Desarrollo de Software

Para Sommerville (2005), el ciclo de vida del desarrollo de software en el conjunto de etapas por las cuales debe transitar el proceso que permite obtener como resultado una solución informática que sea capaz de atender un objetivo determinado. La figura N° 5 sintetiza este ciclo.

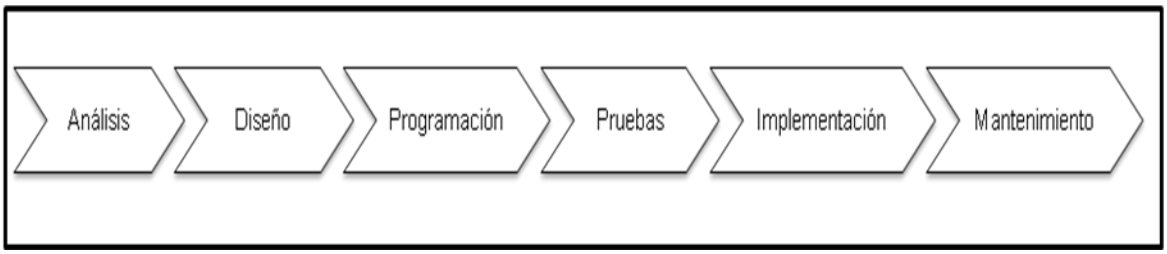


Figura N° 5. Ciclo de vida del desarrollo de software.
Fuente: Sommerville (2005)

Sommerville (2005) define estas fases de la siguiente forma. La fase de análisis tiene por objetivo comprender cuál es la situación para la que se requiere el software, la fase de diseño tiene como objetivo producir el conjunto de especificaciones con las que se programará el software, la fase de programación tiene por objetivo desarrollar todos los programas que conformarán el software, la fase de pruebas tiene por objetivo realizar las pruebas de calidad y aceptación de los programas que conforman el software, la fase de implementación tiene por objetivo poner en funcionamiento el software y la fase de mantenimiento tiene por objetivo realizar las actualizaciones necesarias para permitir que el software siga en operación.

Metodologías de Desarrollo de Software

La ingeniería de software dispone de una variedad de herramientas llamadas métodos de desarrollo de software, las cuales tienen la finalidad de proveer un marco estructurado con el cual se pueda llevar a cabo la tarea de diseñar y construir aplicaciones informáticas. Sommerville (2005) resume las distintas metodologías propuestas para alcanzar este fin; partiendo desde el análisis y diseño estructurado de sistemas de los años 70 y 80, métodos de desarrollo concurrente, métodos de desarrollo formal, métodos incrementales tales como desarrollo de rápido de aplicaciones, desarrollo en espiral, desarrollo de prototipos, método Proceso Unificado de Rational (RUP). La escogencia de uno de estos métodos ya sea de forma pura o mixta, dependerá del proyecto de desarrollo del software que se desee iniciar. En su artículo Gacitúa (2003)

concluye que no existe un acuerdo en cuál es el mejor método a usar y que dependerá del tipo de proyecto así como la experiencia que el equipo de desarrollo tenga.

Modelo en Cascada

El modelo en cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con las especificaciones de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado.

El modelo de la cascada es el paradigma más antiguo de la ingeniería del software. (Pressman, 2010).

Modelo del proceso Incremental

Combina elementos de los flujos de proceso lineal y paralelo, el modelo incremental aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades, cada secuencia lineal produce incrementos de software susceptibles de entregarse de manera parecida a los incrementos producido en un flujo de proceso evolutivo.

Cuando se utiliza un modelo incremental, es frecuente que el primer incremento sea el producto fundamental. Es decir, se abordan los requerimientos básicos, pero no se proporcionan muchas características suplementarias. El cliente usa el producto fundamental (o lo somete a una evaluación detallada), como el resultado del uso y/o evaluación, se desarrolla un plan para el incremento que sigue. El plan incluye la modificación del producto fundamentalmente para cumplir mejor las necesidades de los clientes, así como la entrega de características adicionales y más funcionabilidad. Este proceso se repite después de entregar cada incremento, hasta terminar el producto final.

El modelo de proceso incremental se centra en que en cada incremento se entrega un producto que ya opera. Los primeros incrementos son versiones

desnudas del producto final, pero proporciona capacidad que sirve al usuario y también le dan una plataforma de evaluación.

El desarrollo incremental es útil en particular cuando no se dispone de personal para la implementación completa del proyecto en el plazo establecido por el negocio. (Pressman, 2010).

Modelo en Espiral

Propuesto en primer lugar por Barry Boehm, el modelo en espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas. Boehm describe el modelo de modo siguiente:

> "…El modelo de desarrollo espiral es un generador de modelo de proceso impulsado por el riesgo que se usa para guiar la ingeniería concurrente por participantes múltiples de sistemas intensivos de software. Tiene dos características distintivas principales. La primera es el enfoque cíclico para el crecimiento incremental del grado de definición de un sistema y su implementación, mientras que disminuye su grado de riesgo. La otra es un conjunto de puntos de referencia de anclaje puntual para asegurar el compromiso del participante con soluciones factibles y mutuamente satisfactorias…" (pag.98 Ob. Cit.).

Ingeniería de Requerimientos

Los requerimientos para una solución informática constituyen el conjunto de características, especificaciones y expectativas que los usuarios de esta necesitan estén contenidas en el producto final. La ingeniería de requerimientos constituye una rama de la ingeniería de software, la cual tiene por finalidad entender y documentar las características que una solución informática debe tener, expresadas desde el punto de vista de las personas que la utilizarán. Para Sommerville (2005) "el proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se denomina ingeniería de requerimientos" (p. 106), este

proceso se descompone en cuatro actividades: estudio de factibilidad del sistema, obtención y análisis de requerimientos, especificación y documentación de requerimientos y validación y aprobación de estos requerimientos.

Se observa que este concepto es fundamental dentro del proceso de elaboración de un plan para el desarrollo de una solución informática.

Software para Gestión de Proyectos

El PMI (2013) hace énfasis en la necesidad de disponer de herramientas que puedan influir en el éxito del proyecto, y dentro de los factores ambientales de la empresa se puede destacar el siguiente, “sistemas de información de gestión de proyectos (p.ej., herramientas automáticas, como una herramienta de software para definir cronogramas, un sistema de gestión de configuración, un sistema de recopilación y distribución de información o interfaces Web a otros sistemas en línea)”

En cada una de las áreas de conocimiento en el PMI (2013) se hace referencia a la utilización de herramientas de software para poder llevar a cabo los procesos que las conforman de manera más eficiente y permitir que los procesos asociados al control puedan realizarse de forma efectiva.

Como se ha podido comprobar, las soluciones de mayor difusión en el mercado no cuentan con soporte para las diez áreas de conocimiento propuestas por el PMI (2013).

Gido y Clements (2008) sintetizan de forma muy clara cuáles deben ser las principales características que debe tener un software para gestión de proyectos, identificándose las siguientes: elaboración de presupuestos y control de costos, calendarios, capacidades de Internet, gráficas y tablas, importación/exportación de datos, manejo de proyectos múltiples y subproyectos, gestión de informes, administración de recursos, planeación y monitoreo y seguimiento de proyectos, programación y seguridad.

Pruebas de Software

Según Pressman (2010), las pruebas de software proporcionan una guía que describe los pasos que deben realizarse como parte de la prueba, cuando se planean y se llevan a cabo dichos pasos, y cuanto esfuerzo, tiempo y recursos se requerirán.

El software se prueba para descubrir errores que se cometieron de manera inadvertida conforme se diseñó y construyó. Con frecuencia, la prueba requiere más esfuerzo que cualquiera otra acción de ingeniería de software.

Las pruebas es un conjunto de actividades que pueden planearse por adelantado y realizarse de manera sistemática. Por esta razón, durante el proceso de software, debe definirse una plantilla para la prueba del software: un conjunto de pasos que incluye métodos de prueba y técnicas de diseño de casos de prueba específicos. Las prueba de software debe incluir prueba de bajo nivel, que son necesarias para verificar que un pequeño segmento de código fuente se implementó correctamente, así como pruebas de alto nivel, que validan las principales funciones del sistema a partir de los requerimientos del cliente.

Pruebas de Integración

Según Pressman (2010), es una técnica sistemática para construir la arquitectura del software mientras se llevan a cabo pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar los componentes probados de manera individual y construir una estructura de programa que se haya dictado por diseño.

En la integración incremental el programa se construye y se prueba en pequeños incrementos, donde los errores son más fáciles de aislar y corregir; las interfaces tienen más posibilidades de probarse completo; y puede aplicarse un enfoque de prueba sistemático.

Pruebas de Validación

Pressman (2010), indica que las pruebas de validación comienzan en la culminación de las pruebas de integración, cuando se ejercitaron componentes individuales, el software está completamente ensamblado como un paquete y los errores de interfaz se descubrieron y corrigieron. En el nivel de validación o de sistema, desaparece la distinción entre software convencional, software orientado

a objetos. Las pruebas se enfocan en las acciones visibles para el usuario y las salidas del sistema reconocibles por el usuario.

La validación puede definirse en muchas formas, pero una definición simple (aunque dura) es que la validación es exitosa cuando el software funciona en una forma que cumpla con las expectativas razonables del cliente.

La validación del software se logra a través de una serie de pruebas que demuestra conformidad con los requerimientos. Un plan de prueba subraya las clases de pruebas que se van a realizar y un procedimiento de prueba define casos de prueba específicos que se diseñan para garantizar que: se satisfacen todos los requerimientos de funcionamiento, se logran todas las características de comportamiento, todo el contenido es preciso y se presenta de manera adecuada, se logran todos los requerimientos de rendimiento, la documentación es correcta y se satisfacen la facilidad de uso y otros requerimientos (por ejemplo, transportabilidad, compatibilidad, recuperación de error, mantenimiento).

Después de realizar cada caso de prueba de validación, existen dos posibles condiciones:

1. La característica de función o rendimiento se conforma de acuerdo con las especificaciones y se acepta.
2. Se descubre una desviación de la especificación y se crea una lista de deficiencias. Las deficiencias o errores descubiertos en esta etapa en un proyecto rara vez pueden corregirse antes de la entrega calendarizada. Con frecuencia es necesario negociar con cliente para establecer un método para resolver deficiencias.

Pruebas de Esfuerzo

Según Pressman (2010), las pruebas de esfuerzo se diseñan para enfrentar los programas con situaciones anormales, ejecuta un sistema en forma que demanda recursos en cantidad, frecuencia o volumen anormales. Por ejemplo, pueden:

1.- Diseñarse pruebas especiales que generen diez interrupciones por segundo, cuando una o dos es la tasa promedio.
2.- Aumentarse las tasas de entrada de datos en un orden de magnitud para determinar cómo responderán las funciones de entrada.
3.- Ejecutarse casos de prueba que requieran memoria máxima y otros recursos.
4.- Diseñarse casos de prueba que puedan causar thrashing (que es un quebranto del sistema por hiperpaginación) en un sistema operativo virtual.
5.- Crearse casos de prueba que puedan causar búsqueda excesiva por datos residentes en disco. En esencia la persona que realiza la prueba intenta romper el programa.

Una variación de la prueba de esfuerzo es una técnica llamada prueba se sensibilidad. En algunas situaciones (la más común ocurre en algoritmos matemáticos), un rango muy pequeño de datos contenidos dentro de las fronteras de los datos válidos para un programa puede causar un procesamiento externo, e incluso erróneo, o profunda degradación del rendimiento. La prueba se sensibilidad intenta descubrir combinaciones de datos dentro de clases de entrada válidas que puedan causar inestabilidad o procesamiento inadecuado.

Pruebas de Rendimiento

Las pruebas de rendimiento se diseñan para poner a prueba el rendimiento del software en tiempo de corrida, dentro del contexto de un sistema integrado. Las pruebas de rendimiento ocurren a lo largo de todos los pasos del proceso de pruebas. Incluso en el nivel de unidad, puede accederse al rendimiento de un módulo individual conforme re realizan las pruebas. Sin embargo, no es sino hasta que todos los elementos del sistema están plenamente integrados cuando puede determinarse el verdadero rendimiento de un sistema.

Las pruebas de rendimiento con frecuencia se aparean con las pruebas de esfuerzo y por lo general requieren instrumentación de hardware y de Software, es decir, con frecuencia es necesario medir la utilización de los recursos (por ejemplo, ciclos del procesador) en forma meticulosa. La instrumentación externa puede monitorear intervalos de ejecución y eventos de registro (por ejemplo

interrupciones) conforme ocurren, y los muestreos del estado de la máquina de manera regular. Con la instrumentación de un sistema, las personas que realiza las pruebas puede descubrir situaciones que conduzcan a degradación y posibles fallas del sistema, (Pressman, 2010).

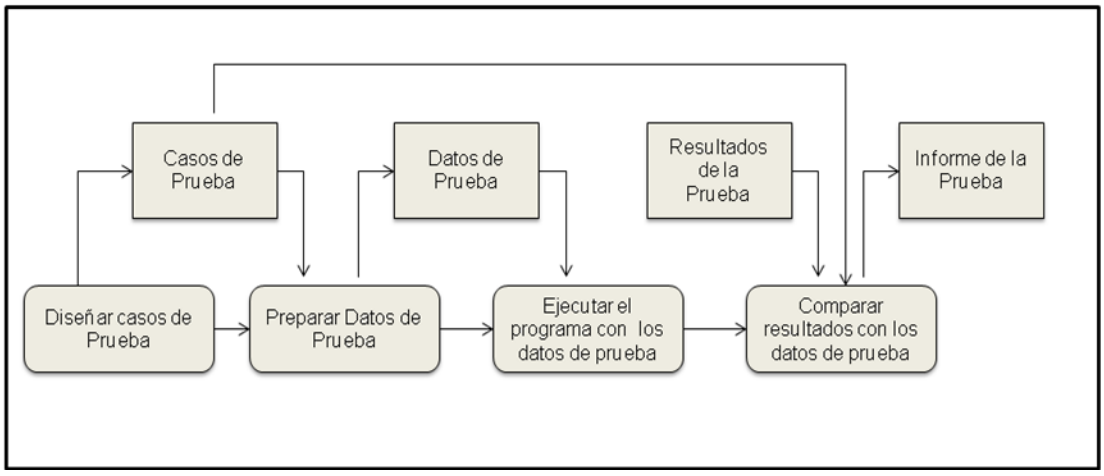


Figura N° 6: Un modelo del proceso de pruebas del software.
Fuente: Sommerville (2005).

Pruebas de aplicaciones WEB

Según Pressman (2010), las pruebas de una página Web (webapp), es una colección de actividades relacionadas con una meta, descubrir errores en el contenido, función, utilidad, navegabilidad, rendimiento, capacidad y seguridad de esa aplicación. Para lograr esto, se aplica una estrategia de prueba que abarca tanto revisiones como pruebas ejecutables.

El proceso de prueba de una webapp comienza enfocándose en los aspectos visibles para el usuario de la aplicación y avanza hacia pruebas que ejercitan la tecnología y la infraestructura. Se realizan siete pasos durante las pruebas: prueba de contenido, prueba de interfaz, prueba de navegación, prueba de componente, prueba de configuración, prueba de rendimiento y prueba de seguridad.

La prueba de contenido (y las revisiones) se enfocan en varias categorías de contenido. La intención es descubrir errores semánticos y sintácticos que afectan

la precisión del contenido o la forma en la que se presenta al usuario final. La prueba de interfaz ejercita los mecanismos de interacción que permiten al usuario comunicarse con la webapp y valida los aspectos estéticos de la interfaz. La intención es descubrir errores que resultan a partir de mecanismos de interacción pobremente implantados o de omisiones, inconsistencias o ambigüedades en la semántica de la interfaz.

Las pruebas de navegación aplican casos de usos, derivado de la actividad de modelado, en el diseño de casos de prueba que ejercitan cada escenario de uso contra el diseño de navegación. Los mecanismos de navegación se ponen a prueba para garantizar que cualquier error que impida completar un caso de uso se identifique y corrija. La prueba de componente ejercita las unidades de contenido y funcionales dentro de la webapp.

La prueba de configuración intenta descubrir errores y/o problemas de compatibilidad que son específicos de un entorno cliente o servidor particular. Entonces se realizan pruebas para descubrir errores asociados con cada posible configuración. Las pruebas de seguridad incorporan una serie de pruebas diseñadas para explotar las vulnerabilidades en la webapp y su entorno. La intención es encontrar huecos de seguridad. La prueba de rendimiento abarca una serie de pruebas que se diseñan para valorar el tiempo de respuesta y la confiabilidad de la misma conforme aumenta la demanda en la capacidad de recursos en el lado del servidor.

Gestión de la Calidad del Proyecto

Se entiende como gestión de la calidad al conjunto de procesos y actividades que tienen por objetivo permitir que el proyecto satisfaga las necesidades por las cuales fue iniciado. Para el PMI (2013) la gestión de la calidad se define como “los procesos y actividades de la organización ejecutante que determinan responsabilidades, objetivos y políticas de calidad a fin de que el proyecto satisfaga las necesidades por la cuales fue emprendido” (p. 227). Para lograr la gestión de la calidad será necesario llevar a cabo los siguientes procesos:

- Planificar la gestión de la calidad es el proceso que permite identificar los requisitos de calidad para el proyecto y documentar como el proyecto demostrará el cumplimiento de los requisitos de calidad.
- Realizar el seguimiento de calidad es el proceso que permite auditar los requisitos de calidad necesarios para el proyecto
- Realizar el control de calidad es el proceso que permite monitorear y registrar de los resultados de la ejecución de las actividades de calidad permitiendo evaluar el desempeño y recomendar cambios necesarios.

Gestión de los Stakeholder del Proyecto

Se entiende como la gestión de los interesados del proyecto al conjunto de procesos que permiten identificar a las personas, grupos u organizaciones que podrían afectar o ser afectados por el proyecto, permitiendo analizar, comunicarse y establecer estrategias en función de los interesados del proyecto. Para el PMI (2013) la gestión de los interesados se define como “los procesos necesarios para identificar a las personas, grupos u organizaciones que podrían afectar o ser afectados por el proyecto, para analizar las expectativas de las partes interesadas y su impacto en el proyecto, y para desarrollar estrategias de gestión adecuadas para la participación efectiva de los interesados en las decisiones y la ejecución de proyectos” (p. 391). Para llevar a cabo la gestión de los interesados será necesario efectuar los siguientes procesos:

- Identificar a los interesados es el proceso que permite identificar a las personas, grupos u organizaciones que podrían afectar o ser afectados por una decisión o resultado del proyecto.
- Planificar la gestión de los interesados es el proceso que permite desarrollar estrategias de gestión que permitan que los interesados participen de manera efectiva durante todo el ciclo de vida del proyecto.
- Gestionar la participación de los interesados es el proceso que permite mantener la comunicación y el trabajo con los interesados del proyecto, fomentando el compromiso de estos con las actividades del proyecto.

- Controlar la participación de los interesados es el proceso que permite mantener el seguimiento de las relaciones con los interesados y las estrategias de ajuste y los planes de participación de los estos.

A continuación la figura N° 7 resume las diez áreas de conocimiento del PMI (2013) con sus respectivos procesos.

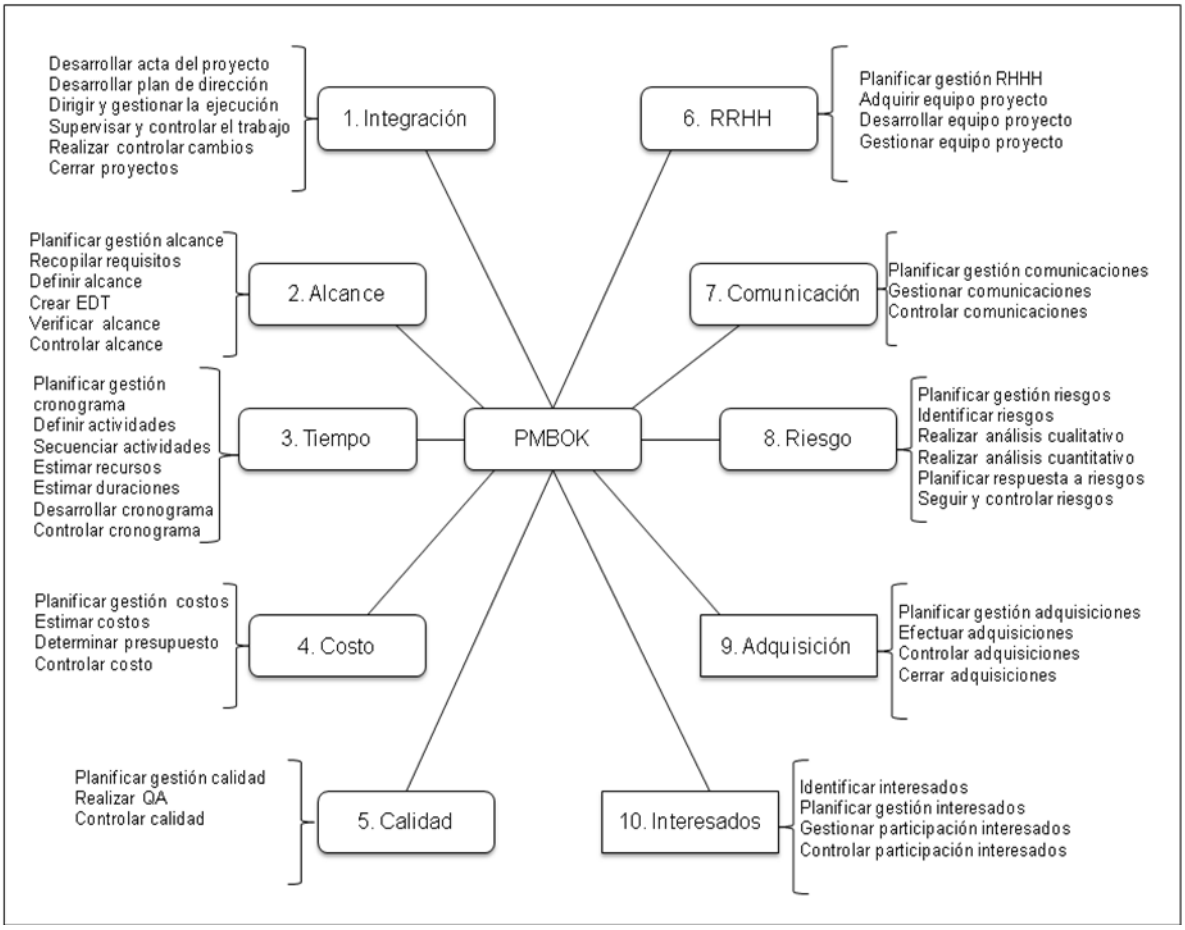


Figura N° 7. Las diez áreas del conocimiento.

Fuente: Adaptado PMI 2013.

Calidad de Servicio y Satisfacción al Cliente

Para definir la calidad de servicio es necesario conocer el concepto de cada una de estas palabras por separado, y así construir una idea de lo que significa. La calidad según lo expresado en la norma FONDONORMA-ISO 9000:2006 es el grado en el que un conjunto de características inherentes cumple con los

requisitos. Está relacionada con el hacer las cosas bien a la primera, con escuchar la voz del cliente y con lograr que los productos entregados al cliente cumplan con las especificaciones técnicas y estándares definidos considerando los deseos de los clientes.
Por su parte en el servicio lo que prevalece es la entrega de un producto intangible al cliente. En el contexto de la norma FONDONORMA-ISO 9000:2006 un servicio es un tipo de producto que resulta de al menos una actividad en la interfaz entre el proveedor y el cliente. Los servicios conforman el sector terciario de la economía y pueden ser la prestación principal de una empresa o aquella que acompaña al producto tangible.

Sin embargo, aunque por separado existen conceptos estándares, para la calidad de servicio no se tiene una definición tan precisa, es más bien de naturaleza difusa y compleja. Así, tratando de generar un concepto a partir de lo expuesto, la calidad de servicio podría verse como el grado en el que las características del producto intangible cumplen con los requisitos del cliente, ya sea interno o externo.

La calidad de servicio se diferencia de la calidad de un bien por ciertas características que lo hace particular y dificulta su medición. Esas palabras que lo identifican son: intangibilidad, heterogeneidad, inseparabilidad y carácter perecedero. Por ser intangible, se dificulta el establecimiento de sus especificaciones y el logro de su estandarización, medición y evaluación. Su heterogeneidad se produce porque es muy variable, depende de la percepción de cada cliente y esto puede variar en cada caso a pesar de ofrecer el mismo tipo de servicio. Es inseparable debido a que en el proceso existe interacción entre el cliente y el proveedor. Es perecedero por que debe consumirse al momento cuando se produce, no puede ser almacenado para ofrecerse luego.

Bajo estas premisas, la calidad de servicio debe ser orientada al cliente, a satisfacer sus expectativas, ya que dependiendo de la perspectiva que forme del producto recibido, será el grado en el que valore su calidad.

Estas consideraciones deben ser aplicadas para el logro de la satisfacción del cliente, que según la norma FONDONORMA-ISO 9000:2006 es la percepción que tiene el cliente sobre el grado en que se han cumplido sus requisitos. Por tal motivo, para una empresa es muy importante conocer los requisitos de los clientes sobre el producto que esperan recibir. Luego, cuando los requisitos están identificados deben transformarse en especificaciones técnicas que formen parte del proceso productivo incluyendo el servicio postventa. Los requisitos del cliente deben ser revisados continuamente por los cambios a los que pueden estar sujetas las necesidades de los clientes. Con el avance de la tecnología, el cliente puede cambiar su percepción acerca de un determinado producto aunque las características del mismo permanezcan iguales.

Comúnmente se define la calidad de servicio como la diferencia entre la percepción del cliente y sus expectativas. Así, depende del cliente. Si su percepción del producto recibido supera o iguala las expectativas que tenía formada, entonces el grado de calidad será alto. Si por el contrario, el cliente percibe que sus expectativas no fueron cubiertas, considerará que el servicio recibido fue de baja calidad.

2.3. Bases Legales

A continuación se presentaron los aspectos legales y normativas del Estado Venezolano, que regula los asuntos en materia de empresas.

Constitución de la República Bolivariana de Venezuela (1999).

Establece en el Artículo 110 que el Estado reconocerá el interés público de la ciencia, la tecnología, el conocimiento, la innovación y sus aplicaciones y los servicios de información necesarios por ser instrumentos fundamentales para el desarrollo económico, social y político del país, así como para la seguridad y soberanía nacional. Para el fomento y desarrollo de esas actividades, el Estado destinará recursos suficientes y creará el sistema nacional de ciencia y tecnología de acuerdo con la ley. El sector privado deberá aportar recursos para las mismas. El Estado garantizará el cumplimiento de los principios éticos y legales que deben

regir las actividades de investigación científica, humanística y tecnológica. La ley determinará los modos y medios para dar cumplimiento a esta garantía.

El Artículo 117, que todas las personas tendrán derecho a disponer de bienes y servicios de calidad, así como a una información adecuada y no engañosa sobre el contenido y características de los productos y servicios que consumen, a la libertad de elección y a un trato equitativo y digno. La ley establecerá los mecanismos necesarios para garantizar esos derechos, las normas de control de calidad y cantidad de bienes y servicios, los procedimientos de defensa del público consumidor, el resarcimiento de los daños ocasionados y las sanciones correspondientes por la violación de estos derechos. Y en el artículo 156 numeral 17, donde especifica que el régimen de metrología legal y control de calidad es de la competencia del Poder Público Nacional.

Ley Orgánica de Ciencia, Tecnología e Innovación (LOC TI).

Según la Gaceta Oficial de la República Bolivariana de Venezuela, N° 39.575, de fecha 16 de diciembre de 2010, promulga, reconocer el funcionamiento de la Ley y sus respectivos reglamentos en la promoción, estimulo y fomento de la investigación, apropiación social del conocimiento, la transferencia e innovación tecnológica, a través de los proyectos que enuncian los beneficiarios.

La normativa legal especificada anteriormente y otros instrumentos legales que podrían aplicar para el desarrollo del modelo objeto de esta investigación, se resumen en la tabla N° 4.

Tabla N° 4. Bases Legales a considerar.

Instrumentos Legales	Artículos Aplicables	Observaciones
Constitución de la República Bolivariana de Venezuela (Gaceta Oficial Extraordinaria N°5.453 de fecha 24/03/2000)	Articulo 117 y articulo 156 numeral 17	Se dispone que las personas tengan derecho a disponer de bienes y servicios de calidad y se establece que el régimen de metrología legal y control de calidad es de la competencia del Poder Público Nacional.
Constitución de la República Bolivariana de Venezuela (año 1999)	Articulo 110	Reconocerá el interés de las personas a la ciencia, la tecnología, el conocimiento, la innovación y sus aplicaciones y los servicios de información necesarios, la ley establecerá los modos y medios para dar cumplimiento a esta garantía.
Ley del sistema Venezolano para la Calidad (Gaceta oficial N° 37.543, de fecha 07/10/2002)	Todos	Establece los mecanismos necesarios para garantizar los derechos de las personas a disponer de bienes y servicios de calidad en el país.
Ley de Reforma Parcial de la Ley Orgánica de Ciencia, Tecnología e Innovación (Gaceta Oficial No. 39.575 de fecha 16/12/2010).	Todos (si la empresa es sujeto pasivo según lo dispuesto en la Ley)	Esta Ley tiene como objeto dirigir la generación de la ciencia, tecnología, innovación y sus aplicaciones en el país. Es aplicable a las empresas cuyos ingresos brutos anuales, superen las cien mil unidades tributarias (100.000 U.T).

CAPITULO III: MARCO METODOLOGICO

En este capítulo se presentó la metodología a emplear en el desarrollo del presente estudio. Se detallan aspectos como el tipo de investigación, diseño de la investigación, unidad de análisis, población y muestra, técnicas de instrumentos recolección de datos, fases de la investigación , procedimiento por objetivos, operacionalización de los objetivos, estructura desagregada de trabajo, aspectos éticos, cronograma, recursos.

Para abordar de manera científica la realidad tratando de evitar las distorsiones por factores objetivos y subjetivos que dificulten y perturben el conocer, se requiere de métodos que permitan una mejor utilización de los medios para acceder al conocimiento, fijar con antelación y evaluar los resultados de la acción; considerándolos como un modo de aproximación y no como un conjunto de certezas convincentes, ya sea relación al conocimiento o en función de las acciones concretas realizadas.

3.1 Tipo de Investigación

De acuerdo a Palella y Martins (2006) “el tipo de investigación se refiere a la clase de estudio que se va a realizar. Orienta sobre la finalidad general del estudio y sobre la manera de recoger las informaciones o datos necesarios” (p.97).

El Consejo General de Estudios de PostGrado de la Universidad Católica Andrés Bello, en reforma parcial al Reglamento General de los Estudios de Postgrado (2010), establece en su artículo dos (2) lo siguiente:

> El trabajo especial de grado se concibe dentro de la modalidad de investigación cuyo objetivo fundamental es el de aportar soluciones a los problemas y satisfacer necesidades teóricas o prácticas, ya sean profesionales, de una institución o de un grupo social. Se pretende que el alumno demuestre el dominio instrumental de los conocimientos aprendidos en la especialización para lo cual el tema elegido por el estudiante deberá insertarse en una de las materias del plan de estudios correspondientes (p. 1).

El objetivo del presente trabajo es el referido al Diseño un plan de Gestión de Pruebas para los Proyectos de Software para en un futuro contar con una herramienta optima con la calidad que debe estar presente en el desarrollo de los software, para el trabajo en estudio se incorpora el tipo de investigación denominado tipo de **investigación aplicada**.

Según Vieytes (2004), la investigación aplicada se nutre de la investigación básica para resolver problemas concretos y en tal sentido depende de sus logros. La investigación investiga cómo aplicar las leyes.

Es preciso acotar que la investigación aplicada está orientada a conocer las necesidades que no están siendo satisfechas, se orienta hacia la solución de un problema particular. Tiene por finalidad la búsqueda y consolidación del saber y la aplicación de los conocimientos para el enriquecimiento del acervo cultural y científico, así como la producción de tecnología al servicio del desarrollo integral de las naciones.

3.2 Diseño de la Investigación

Para Hurtado (2010) "El diseño se refiere a donde y cuando se recopila la información, así como la amplitud de la información a recopilar, de modo que se pueda dar respuesta a la pregunta de investigación de la forma más idónea posible. " (p.147).

El tipo de investigación se considera No Experimental, ya que no se construye ninguna situación, sino que se observan situaciones ya existentes con la finalidad de analizarlos para luego tomar las acciones que permita mejorar el proceso de pruebas del software.

Igualmente el estudio se caracteriza por ser una investigación documental, ya que se basa en la revisión y análisis de documentos para extraer datos necesarios requeridos en el diseño de un plan de Gestión de Pruebas para los Proyectos de Software, lo cual se apoya en la definición de investigación documental, en donde Eyssautier (2008) indica que: "La investigación documental es una investigación que se efectúa a través de consultas en los documentos, pudiendo ser revistas,

libros, diarios, informes, anuarios o cualquier otro registro que da testimonio de un hecho o fenómeno".(p.116).

Es preciso señalar que el diseño de investigación documental requiere un gran nivel de creatividad y originalidad, además de una gran capacidad de análisis, síntesis y reflexión.

3.3 Unidad de Análisis

La unidad de análisis corresponde con los sujetos u objetos de estudio, que Hurtado (2010) define como "las entidades (personas, objetos, regiones, instrucciones, documentos, plantas, animales, productos), que poseen el evento de estudio". (p.140). De aquí se desprende que la unidad de análisis para este trabajo de investigación es el conjunto de software que se analizan para diseñar el plan de gestión de pruebas de software.

3.4 Técnicas de e Instrumentos Recolección de Datos

A continuación se presentan las técnicas e instrumentos de recolección de información que se emplearán en la investigación, en función de los objetivos definidos

De acuerdo a lo señalado por Arias (2012), "se entenderá por técnica, el procedimiento o forma particular de obtener datos o información", siendo estas técnicas diversos procedimientos, forma o maneras que conformarán los criterios que permitirán obtener información, datos u opiniones de la población sobre el tema que se está tratando. Identifica dos técnicas para la investigación documental: análisis documental y análisis de contenido, para la primera identifica dos instrumentos de recolección de datos: ficha y computadora y para la segunda un instrumento de recolección de datos: cuadro de registro y clasificación de categorías.

Para Hurtado (2010) las técnicas de recolección "tienen que ver con los procedimientos utilizados para la recolección de datos, es decir, el cómo. Estas

pueden ser de revisión documental, observación, encuesta y técnicas sociometrías, entre otras, (p. 153).

Es preciso acotar que uno de los principales instrumentos de recolección y análisis de la información en este estudio fue el investigador, quien es el responsable de obtener los datos y de hacer las relaciones entre los aspectos observados. Es quien tiene y desarrolla la experticia en el tema objeto de análisis.

Las técnicas de recolección de datos que se utilizaran en el estudio al que se hace referencia en esta investigación son las siguientes:

1.- Análisis documental

Para Eyssautier (2008) el análisis documental "es aquella que depende exclusivamente de fuentes de datos secundarios, o sea, aquella información que existe en documentos y material de índole permanente". (p.159)

2.- Observación

Valarino, Yaber, y Cemborain (2011) definen la observación "como la acción de percibir un fenómeno a través de los sentidos o por medio de aparatos." (p.218), por otra parte Evans y William (2008) definen la observación como, "son tipos especiales de formas para recopilar datos en las cuales los resultados se pueden interpretar directamente sobre la forma, sin necesidad de un procedimiento adicional". (p.699).

3.- Juicio de Expertos

El juicio de expertos es otra técnica relevante que se utilizara para este tipo de investigación, que, según Escobar y Cuervo (2008), "se define como una opinión informada de personas con trayectoria en el tema, que son reconocidas por otros como expertos cualificados en este, y que pueden dar información, evidencia, juicios y valoraciones". (p. 29).

Los documentos y registros, permitirán al investigador entender el ambiente donde se realizara el estudio y servirá de referencia para analizar los datos.

3.5 Fases de la Investigación

Se definirán cuatro fases que favorecerán el logro de los objetivos planteados.
Estas fases se desarrollaran para caracterizar, analizar, diagnosticar e identificar los elementos necesarios en el proceso de pruebas de un software, que permitan formular un plan de Gestión de Pruebas para los Proyectos de Software.

3.5.1 Fase I Levantamiento de la Información:

Esta fase tiene como objetivo elaborar un análisis de las características funcionales que presentara el producto, haciendo una revisión de los manuales de uso, de usuario, para tener base de la información, así como el empleo de las aplicaciones de tal forma de identificar las características que deben ser consideradas en el plan a diseñar.

3.5.2 Fase II Conceptualizacion:

El objetivo de esta fase es identificar y definir las características funcionales que deben ser consideradas como requisito dentro del plan de gestión de pruebas para los proyectos de software a realizar, así como los aspectos propios delineados por el PMI.

3.5.3 Fase III Diseño y Desarrollo:

En esta fase se desarrollan los procesos de planificación de las áreas de desarrollo del software, tomadas en cuenta para esta investigación.

3.5.4 Fase IV Informe Final:

Esta última fase involucra la formulación de las fases del plan de gestión de pruebas para los proyectos de software siguiendo los lineamientos de la ingeniería del software y las buenas prácticas de la gerencia de proyecto, en este sentido se establecerán los documentos formales que permitan el diseño del plan de gestión.

3.6 Operacionalización de las Variables

En esta sección se definen, se descomponen y delimitan las variables a estudiar en este trabajo, para visualizar los aspectos de la realidad que serán investigados. De acuerdo con Arias (2012), variable “es una característica, cualidad o medida que puede sufrir cambios y que es objeto de análisis, medición o control en una investigación” (p.55); por lo que la Operacionalización de una variable significa establecer el proceso mediante el cual se transforma la variable de conceptos abstractos a términos concretos, observables y medibles, es decir, dimensiones e indicadores.

La definición operacional es un “conjunto de procedimientos y actividades que se desarrollan para medir una variable” (Hernández et al., 2010, p.111). Este autor también indica que “los criterios para evaluar una definición operacional son básicamente cuatro: adecuación al contexto, capacidad para captar los componentes de la variable de interés, confiabilidad y validez.” (p. 112).

Hurtado (2010), define la operacionalización como “el proceso que le permite al investigador identificar aquellos aspectos perceptibles de un evento, que hace posible dar cuenta de la presencia o intensidad de éste” (p. 131).

Se trata de descomponer, luego de su definición nominal y conceptual, cada una de las variables en estudio en los aspectos que la componen, a fin de facilitar la recolección con alto grado de precisión, de los datos necesarios. A continuación en la Tabla N° 5, se desglosan las variables objeto de estudio.

Tabla N° 5. Operacionalización de las Variables.

Evento: Diseñar un plan de Gestión de Pruebas para los Proyectos de Software.			
Objetivos Específicos	**Dimensión**	**Instrumento**	**Fuente de Información**
Caracterizar los proyectos de Software de la empresa en estudio.	Especificaciones a considerar en el plan. Requerimientos requisitos	Revisión Documental Metodología de diseño de software Aplicaciones comunes	Diseño de Software PMI (2013) PDRI
Analizar las mejores prácticas en el área de gestión de pruebas para proyectos de Software.	Definición previa de Productos finales Implementar ambiente de pruebas Implementar métricas	Plan de pruebas Requerimientos Casos de prueba Número total de componentes del producto Número de componentes con errores	Diseño de Software PMI (2013) Metodología desarrollo de Software
Formular una propuesta basada en las mejores prácticas de proyectos de software.	Plan de desarrollo de proyectos	Investigación Documental	Diseño de Software PMI (2013) Metodología desarrollo de Software Metodología de casos de prueba
Elaborar las fases del plan de gestión de pruebas para la empresa objeto de estudio.	Plan de desarrollo de proyectos	Investigación Documental	Testers

3.7 Estructura Desagregada de Trabajo (EDT/WBS)

Desde el punto de vista de Gray & Larson (2009) la EDT" es un diagrama esencial del proyecto con distintos niveles de detalle. También se puede utilizar para definir los canales de comunicación y ayudar a comprender y coordinar muchas partes del proyecto". (p. 91). En este mismo orden de ideas el PMI (2013) señala que "el beneficio clave de este proceso, es que proporciona una visión estructurada de lo que tiene que ser entregado." (p. 125). En la figura N° 8, se presenta la WBS donde se describen las actividades que comprenden el alcance de esta investigación.

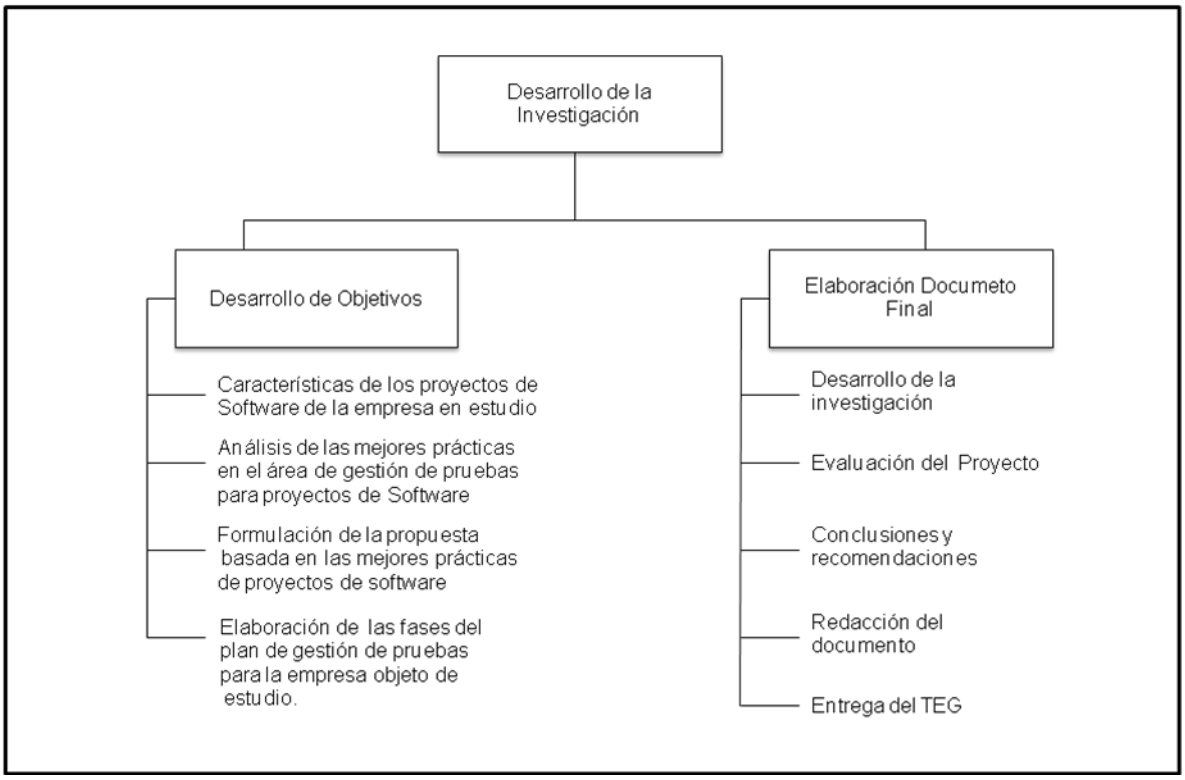


Figura N° 8. Estructura Desagregada de Trabajo de investigación.

3.8 Aspectos Éticos

Este trabajo de investigación está apegado al código de ética del Colegio de Ingenieros de Venezuela y al Código de Ética y Conducta Profesional del PMI.

3.8.1 Código de Ética Colegio del Ingeniero de Venezuela (CIV)

La ingeniería es una profesión importante a la que se llega mediante un concierto de valores, conocimientos científicos y tecnológicos, que con la técnica y el arte analiza, crea y desarrolla sistemas, productos, procesos y obras físicas, mediante el empleo de la energía y materiales para proporcionar a la humanidad con eficiencia y sobre bases económicas, bienes y servicios que le den bienestar con seguridad y creciente calidad de vida preservando el medio ambiente. Tiene un impacto en la sociedad ya que a través de ella se prestas un servicio que requiere honradez, honestidad, integridad, imparcialidad entre otros valores de los ingenieros.

Valarino, Yaber, y Cemborain (2011), agrega que el investigador debe “ser fiel a los resultados obtenidos y decir la verdad, sin falsear los mismos a su conveniencia, además de no apropiarse de las ideas ajenas, ya sea de los escritos o de las investigaciones previas.”

El desarrollo de esta investigación, se realizará dando cumplimiento a las leyes nacionales e internacionales inherentes al tema, así como al código de ética de la sociedad profesional relacionada con el investigador, que en este caso, es el Colegio de Ingenieros de Venezuela.

Los códigos de ética del Colegio de Ingenieros a los cuales se apega la rectitud de esta investigación son:

- **Primero (virtudes):** Actuar en cualquier forma que tienda a menoscabar el honor, la responsabilidad y aquellas virtudes de honestidad, integridad y veracidad que deben servir de base a un ejercicio cabal de la profesión.
- **Segundo (ilegalidad):** Violar o permitir que se violen las leyes, ordenanzas y reglamentaciones relacionadas con el cabal ejercicio profesional.

- **Tercero (conocimiento):** Descuidar el mantenimiento y mejora de sus conocimientos técnicos, desmereciendo así la confianza que al ejercicio profesional concede la sociedad.
- **Cuarto (seriedad):** Ofrecerse para el desempeño de especialidades y funciones para las cuales no tengan capacidad, preparación y experiencias razonables
- **Octavo (firma):** Firmar inconsultamente planos elaborados por otros y hacerse responsable de proyectos o trabajos que no están bajo su inmediata dirección, revisión o supervisión.
- **Décimo Octavo (autoría):** Utilizar estudios, proyectos, planos, informes u otros documentos, que no sean el dominio público, sin la autorización de sus autores y/o propietarios.
- **Décimo Noveno (secreto):** Revelar datos reservados de índole técnico, financiero o profesionales, así como divulgar sin la debida autorización, procedimientos, procesos o características de equipos protegido por patentes o contratos que establezcan las obligaciones de guardas de secreto profesional. Así como utilizar programas, discos, cintas u otros medios de información, que no sea de dominio público, sin la debida autorización de sus autores y/o propietarios, o utilizar sin autorización de códigos de acceso de otras personas, en provecho propio.

  En resumen de las 22 disposiciones que allí se exponen, se debe actuar de forma que se respeten virtudes como la honestidad, integridad y veracidad; siempre cumpliendo con el marco legal a pesar de la amistad, conveniencia o coacción; manteniendo y mejorando los conocimientos; ejerciendo funciones para las cuales tengan la capacidad, preparación y experiencias mínimas requeridas; respetando las remuneraciones mínimas establecidas por el Colegio de Ingeniero de Venezuela.

  En cuanto a los proyectos o informes, evitar la negligencia; el hacerse cargo de trabajos dirigidos, supervisados o revisados por otros; el comenzar sin realizar todos los estudios previos necesarios; el concurrir deliberadamente o invitar a licitaciones o el ejercer influencias para crear situaciones de privilegio. La información que no sea de dominio público debe ser utilizada con la autorización

de sus autores y/o propietarios, no se revelará información confidencial, no se deben someter a los clientes a la experimentación o a servicios no necesarios, ni se hará publicidad indebida.

No se deben usar ventajas inherentes a un cargo remunerado para competir con la práctica independiente de otros profesionales; ni se atentará contra la reputación de otros. Se evitará el conflicto de intereses con los involucrados en la investigación; manteniendo los principios de justicia y lealtad en dicha relación. Se respetará el ambiente, se facilitará la contratación de profesionales nacionales y se cumplirá con lo dispuesto en las “Normas de Actuación Gremial del CIV”.

3.8.2 Código de Ética y Conducta Profesional PMI

El propósito de este Código es infundir confianza en el ámbito de la dirección de proyectos y ayudar a las personas a ser mejores profesionales. Para ello, establece el marco para entender los comportamientos apropiados en la profesión. Creemos que la credibilidad y reputación de la dirección de proyectos como profesión se forjan sobre la base de la conducta colectiva de cada profesional.

- **Capítulo II:** Responsabilidad.
- **Capítulo III:** Respeto.
- **Capítulo IV:** Equidad.
- **Capítulo V:** Honestidad.

Los valores anteriormente descritos se conforman en los pilares sobre los cuales se considerará el correcto comportamiento de los integrantes de quienes fundamentan su quehacer profesional sobre las mejores prácticas promovidas por el PMI (2013)

3.9 Cronograma

El cronograma de actividades para la investigación en estudio, es una herramienta grafica que muestra las actividades realizadas en la investigación y el lapso de tiempo que cada una de ellas requiere para ser completada. El cronograma es una herramienta de planificación y control de proyectos, con una utilidad significativa en investigación. Hurtado (2010), considera el cronograma una parte importante

del proceso de planificación, el cual debe contener las actividades así como la duración en tiempo de éstas.

Con la utilización de la herramienta Microsoft Project se muestra a través de la figura N° 9, el diagrama de GANTT, las diferentes tareas o actividades que se plasman en la estructura desagregada de trabajo mencionada con el fin de cumplir los objetivos planteados.

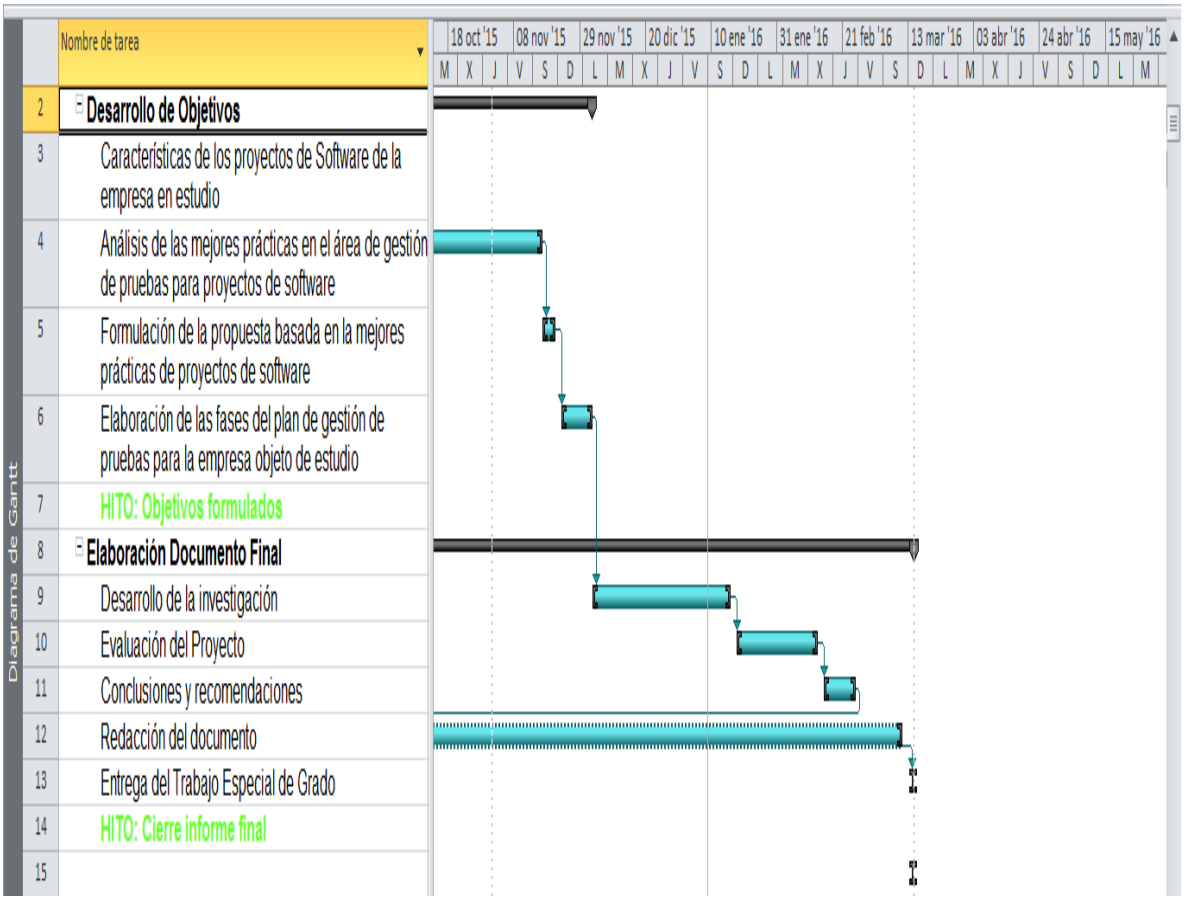


Figura N° 9. Cronograma del Proyecto de investigación.

3.10 Recursos

De acuerdo a lo mencionado por Valarino (2011) y otros, los recursos:

Se refieren a los recursos humanos o personas que participaran en la investigación: recursos tecnológicos y materiales, como software, maquinas,

procesos, computadoras, equipo, además de los recursos financieros que, por lo general, en un trabajo de investigación aplicada son aportados por el investigador y por la empresa donde se va a realizar la investigación. (p. 2010).

Para el desarrollo de la presente investigación se estimaron los siguientes recursos:

Tabla N° 6. Presupuesto Proyecto de Investigación.

Cantidad/ Horas	Tipo de Recurso	Descripción	Costo Unitario BsF	Estimación Costo (BsF)
15	Recursos Humanos	Asesor	600,00	9.000,00
80	Recursos Humanos	Estudiante	150,00	12.000,00
200	Consumibles	Copias blanco y negro	30,00	6.000,00
30		Copias a color	40,00	1.200,00
4		CD	1.250,00	5.000,00
1		Tinta	30.000,00	30.000,00
1		Tóner	40.000,00	40.000,00
3	Inscripción Seminario	Seminario de Grado	1.442,00	4.326,00
4,8	Obtención del Titulo	Inscripción del TEG	1.441,88	6.921,00
1		Derecho de Grado	1.500,00	1.500,00
		Total Bs.		**115.947,00**

Cuyo presupuesto asciende a un costo aproximado de: 115.947 Bs. para el desarrollo satisfactorio del Trabajo Especial de Grado.

CAPITULO IV: VENTANA DE MERCADO

La Industria Venezolana del Software (IVS) es un sector de la economía nacional en pleno proceso de crecimiento. Este sector está conformado, en su mayoría, por pequeñas y medianas empresas (PYMES); muchas de las cuales, están tratando de mejorar sus procesos, para aumentar su competitividad y ganar mercados fuera del ámbito nacional.

La calidad del software es considerada un requisito fundamental para competir en el mercado nacional e internacional.

Mercado Potencial

Las oportunidades del mercado son amplias, siempre que ofrezca un software de calidad, además, el cliente siempre busca un proveedor que le proporcione soluciones de manera rápida y eficaz para adaptar sus programas a la actualidad del mercado y adecuarse a las regulaciones y exigencias presentes en el mercado venezolano. Es usual encontrar empresas de alta trayectoria que se encuentren en el proceso de renovación o actualización de su estructura informática y soluciones de software. Con base a las entrevistas de opinión calificada, se establece una duración promedio de 4 años con la misma solución de software, para posteriormente evaluar la pertinencia de efectuar actualizaciones parciales o totales mediante licitaciones para estudiar nuevas propuestas de actuales o nuevos proveedores.

Venezuela es un país comercialmente muy atractivo y constantemente hay nuevas empresas en el mercado con necesidad de adquirir tecnología en el área de software, especialmente vinculados al sector comercio al ser un país netamente importador de bienes no petroleros, para su eficaz funcionamiento con base a sus necesidades específicas de acuerdo a su ámbito de trabajo.

Producto

Los servicios de software consisten en otorgar soluciones informáticas y plataformas de inteligencia a las organizaciones que se dedican principalmente a la venta al por menor, proporcionado las herramientas de negocios más precisas para anticiparse a los problemas y obtener soluciones, dándole capacidad de tomar decisiones acertadas a la empresa para poder lograr aumentar la rentabilidad y ventas.

Las soluciones de software permiten llevar a cabo una o varias tareas específicas, de manera rápida y eficaz, es por ello, que muchas empresas optan por cotizar y adquirir este tipo de servicio con el fin de optimizar su trabajo y desde luego su productividad.

Las líneas de productos, principalmente, ofrecidas por las empresas Industria Venezolana del Software (IVS) son las de desarrollo de aplicaciones hechas a la medida del cliente y bajo plataformas web o cliente-servidor. Las dos líneas de servicios más atendidas son la consultoría y la comercialización/distribución de software.

Herramientas de Construcción de Software

En Venezuela, se está trabajando en varias modalidades de licenciamiento, estos son: libre o privativos. Ellos se corresponden con el cliente a quien está destinado el producto. Las licencias libres son, generalmente, requeridas por clientes gubernamentales; mientras que las privativas son más requeridas por empresas privadas o no gubernamentales. Entre las herramientas que deben ser consideradas dentro de estas modalidades están: Los lenguajes de programación, los sistemas operativos y los sistemas manejadores de base de datos, ellas son fundamentales para el desarrollo de aplicaciones de software.

En relación a este aspecto, se obtuvo que Windows es el sistema operativo más utilizado, seguido de Linux. Sin embargo. Entre los sistema manejadores de base de Datos se tiene que primero esta SQL Server como el más utilizado, seguidos por MySQL, Oracle y PostgreSQL, con respecto a los lenguajes de programación

usados para el desarrollo de aplicaciones, se tienen PHP, Java, Visual Studio (.net).

Promoción de Productos

Actualmente en Venezuela existen buenas proyecciones en términos de demanda y se espera que el mercado de software continúe aumentando conforme las necesidades y expansión comercial y empresarial lo requieran, además las adecuaciones y modernizaciones necesarias que vienen implementando las compañías locales en su plan de desarrollo. La tendencia en tecnologías de información es de un crecimiento promedio al planteado en el mercado global.

El país vive una etapa de adecuación a los nuevos marcos legales, lo cual permite mantener un crecimiento estable de la demanda durante todo el año.

Hoy en día la mayoría de las empresas desarrolladoras permiten obtener sus productos de manera electrónica mediante descargas on-line, sin embargo, en Venezuela la principal forma de comercialización de los servicios es a través de distribuidores especializados autorizados, además de proveedores directos de estas mismas empresas a grandes y medianas empresas que venden los programas necesarios para el funcionamiento cotidiano de las distintas compañías, adaptándose a los diferentes sectores a los cuales se dediquen.

Productores del bien o el servicio

Según información concedida por la Cámara Venezolana de Empresas de Tecnologías de la Información (CAVEDATOS), en el mercado de software se encuentran aproximadamente 280 empresas, de las cuales un aproximado de 120 empresas se dedican a la venta de software de ventas retail; gran proporción de éstas ubicadas en la ciudad de Caracas, estando dedicadas en su mayoría a soluciones empresariales y de negocios.

En Venezuela las inversiones en tecnologías de información por parte de las empresas venezolanas que han adquirido o actualizado nuevos estándares durante el año 2011 –última cifra oficial disponible- alcanzaron un aproximado de 4.700 millones USD, de las cuales el sector software obtuvo un 7% con

inversiones cercanas a USD 350 millones, según información de estudio de mercado realizado por IDC Venezuela, consultora con oficinas en diversos países de América Latina y Estados Unidos dedicada al análisis de mercados de TI y de Telecomunicaciones en la región.

Para efectos de esta investigación se revisaron los portales de las siguientes empresas: www.miprofit.com; www.corporacionsybven.com; www.saintnet.com; www.eniac.com; www.bekesantos.com; www.winledger.com; www.ve.ibm.com; www.megasoft.com.ve; www.tecnocomputacion.com

Consumidores actuales o potenciales

Con relación a los principales clientes de software para ventas se encuentran grandes, medianas y pequeñas empresas involucradas en los sectores de alimentos, salud, construcción, entre otros. Los resultados arrojados por las diferentes visitas realizadas por la Oficina Comercial a principales cadenas de (supermercado, ferretería, farmacia, transporte) con alta trayectoria en Venezuela, indicaron que los puntos más importantes que se toma en consideración a la hora de adquirir un servicio de software es la reputación de la empresa, según referencias de su cartera de clientes. A modo de ejemplo, luego de haber efectuado las investigaciones pertinentes, en Venezuela existen 24 cadenas incluyendo de hipermercados y supermercados con cobertura a nivel nacional, lo que significa un total de 290 tiendas solo en este segmento. También el mercado posee importante cobertura a nivel de banca, empresas de seguros, alimentos, comercio, manufactura, entre otros, sin contar la industria petrolera que es el motor de la economía.

Otro tema de gran importancia que toman en consideración es el servicio de actualización o complementación, que dependiendo de las necesidades de la empresa se puedan requerir ante la compañía de software y por último lo más importante es que la empresa que provee el software ofrezca atención rápida y eficaz ante cualquier consulta del cliente (servicio postventa).

Competidores de Software

En Venezuela los principales proveedores del sector son extranjeros, destacándose las compañías Microsoft, Oracle y SAP, las cuales controlan gran parte del mercado, sin embargo, debido a la gran cantidad de opciones de programación existente para desarrolladores, no es excluyente la entrada al mercado de empresas de todo tipo de servicios ligados a la industria o al comercio.

Los principales servicios otorgados van desde sistemas operativos, hasta aplicaciones especializadas, proporcionando al mercado una gran variedad de programas para todos los usos.

Microsoft como principal empresa a nivel mundial controla gran parte del mercado en diversos sectores, no obstante, es importante señalar la reducida participación de esta empresa en las aplicaciones de control de sistemas y automatización, empresarial e industrial, entre otros que proporcionan mayores oportunidades para empresas que quieran iniciarse en esta área o en cualquier otra.

Segmentos y Estrategias de Penetración de Competidores

Es de hacer como la inversión en publicidad y mercadeo de las compañías sigue jugando un papel importante al momento de comercializar el servicio que se quiere prestar. Actualmente diferentes compañías tanto locales como extranjeras invierten importantes cantidades de recursos en publicidad audiovisual y escrita con el fin de captar potenciales clientes.

De igual forma las empresas crean eventos o forman parte de alguna otra actividad como ferias y encuentros especializados que les proporcionen una plataforma para atraer y afianzar nichos de mercado.

Valores Aproximados de Servicios Provistos u Ofrecidos por Competidores

Las cotizaciones que puedan proporcionar cada compañía van a depender de diferentes factores como tamaño y necesidades de la empresa, tipo de software, locaciones, infraestructura y planta física, entre otros.

Agente Reguladores

Supervisión del Estado como ente regulador y administrador del comercio del país tanto de bienes como de servicios a través de la ejecución de su política comercial.

Tabla N° 7 Foda de Productores de Servicio de Software.

Fortalezas	Oportunidades	Amenazas	Debilidades
Utilización de buenas prácticas y metodologías agiles para el desarrollo de proyectos	Al pasar la crisis las empresas estarán dispuestas a mejorar su tecnología	La crisis mundial y la decisión de los consumidores de ahorrar	Número limitado de proyectos ejecutándose simultáneamente
Capacidad para adaptarse a nuevas tecnologías	Curiosidad de los clientes sobre nuevas tecnologías	Disminución en las ventas en el mercado	Falta de alianzas estratégicas y convenio con los competidores que beneficien
Infraestructura adecuada y equipamiento y tecnología de última generación	Poseer gran variedad de sistemas que puedan satisfacer las necesidades del cliente	Situación económica y política del país	Existen empresas de competencia en desarrollo de software
Excelencia y compromiso con los clientes	La baja creación de competencia por las barreras de entrada	Ingreso de competidores con estructuras de costos menores	La infraestructura posiblemente con el aumento laboral de proyectos nuevos para la empresa no sea la adecuada

CAPITULO V: DESARROLLO DE LOS OBJETIVOS ESPECIFICOS

En este capítulo se describen los aspectos relevantes para el desarrollo de los objetivos específicos que llevaron a la elaboración del plan de gestión, siendo estos los siguientes:

5.1 Objetivo Nro. 1: Caracterizar los proyectos de Software de la empresa en estudio.

Los proyectos de software desarrollados tienen como finalidad de proporcionar soporte a los procesos de negocio de las organizaciones de forma integral, sencilla y flexible, en las áreas como: inventario, compras, ventas y tesorería, Órdenes de Pago, Impresión de Cheques, pedidos, etc., ayudándole a potenciar el funcionamiento del negocio.

El software desarrollado ofrece una poderosa combinación de funcionalidad, plataformas tecnológicas, capacidad de crecimiento y adaptación, así como el respaldo y los servicios adecuados para proveer beneficios tangibles a las empresas.

Cadena de Valor Operativa

La cadena de valor es un modelo teórico que gráfica y permite describir las actividades de una organización para generar valor al cliente final y a la misma empresa. En base a esta definición se dice que una empresa tiene una ventaja competitiva frente a otra cuando es capaz de aumentar el margen (ya sea bajando los costos o aumentando las ventas). Este margen se analiza por supuesto a través de la cadena de valor de Porter (1985).

Cada empresa es un conjunto de actividades que se desempeñan para diseñar, producir, llevar al mercado, entregar y apoyar a sus productos, basado en el autor antes citado.

Para el caso de objeto en estudio, estas actividades son de naturaleza tecnológica y están basados en su cadena de valor, la cual se presenta a continuación en la figura N° 10

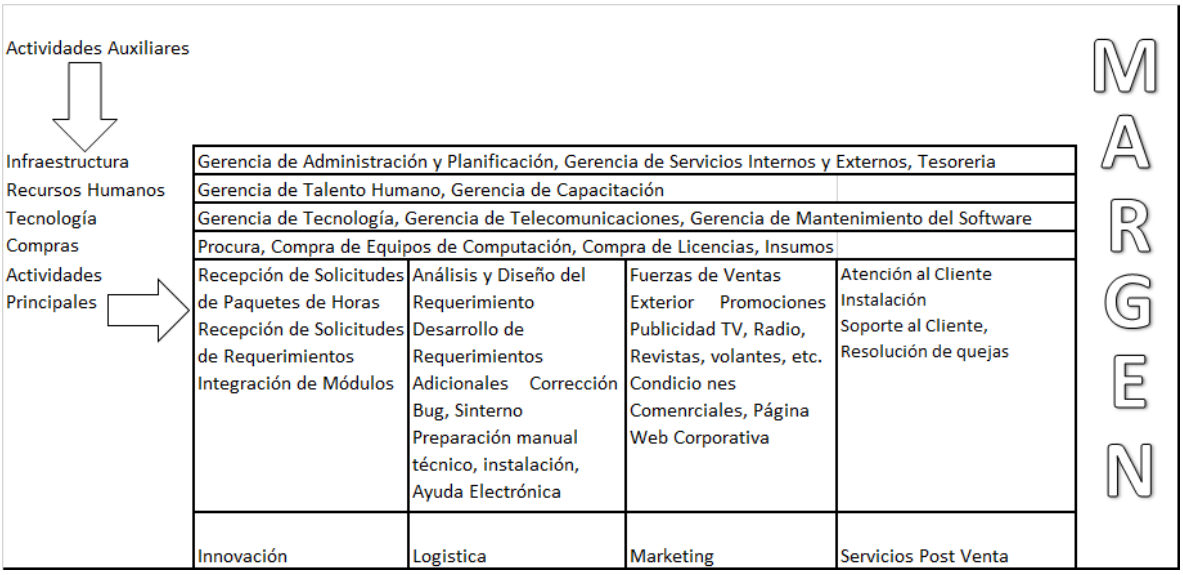


Figura N° 10. Cadena de Valor Operativa.

Fuente: Adaptado de Porter (1985)

Infraestructura de la empresa se desagrega en Gerencia de Administración y Planificación, Gerencia de Servicios Internos y externos, Tesorería, donde se contemplan actividades de planificación, organización, legal, relaciones con reguladores, auditoria, cobranzas y contabilidad.

Las demás actividades auxiliares contemplan:

- ✓ La Gerencia de Talento Humano, permite gestionar la selección, contratación, formar, emplear y retener a los colaboradores de la organización dándoles capacitación y prestándoles una serie de servicios.
- ✓ Gerencia de Tecnología, incluye las actividades del equipo de desarrollo ya que las actividades principales giran en torno al desarrollo de soluciones tecnológicas.
- ✓ Procura, compras, se refiere al abastecimiento de los materiales necesarios en cantidad necesaria, calidad y tiempo requeridos, lo cual será traducido al

mejor servicio al cliente. Incluye la gestión de recursos con los proveedores compra de licencias, equipo de computación de última generación.

Las actividades principales contemplan:

- ✓ Innovación, se refiere a la conversión del conocimiento empresarial en nuevos productos y/o procesos para su introducción en el mercado, lo que permite desarrollar muchas oportunidades, más allá del negocio tradicional de la empresa y crear valor añadido a todos los eslabones de la cadena de la empresa. Entre las actividades primarias de innovación se encuentran el diseño de soluciones de software, soluciones y mejoras adaptados a las medidas y los requerimientos de cliente, el desarrollo y mejoramiento de las técnicas.
- ✓ Logística, comprende la logística interna que corresponde al proceso en donde se agrupan todas las actividades operativas internas de la empresa y la logística externa comprende el almacenamiento de los productos terminados y su distribución a los clientes finales.
- ✓ Marketing, su finalidad es la de reunir los factores y hechos que influyen en el mercado para crear lo que el consumidor requiere, desea y necesita, distribuyéndolo en forma tal, que este a su disposición en el momento oportuno, en el lugar preciso y al precio más adecuado. Comprende las actividades de investigación de mercado y presentaciones a ejecutivos.
- ✓ Servicio Post Venta, se refiere al servicio dado al cliente. Si una empresa logra diferenciarse de la competencia al servicio que ofrece al cliente, está consiguiendo una ventaja competitiva. Destacan las actividades de obtener información al cliente, brindar servicios de apoyo, brindar adiestramientos.

La Suite básica de la familia de productos que se ofrece está conformada por sus sistemas Administrativo, Contabilidad y Nomina, a través de ellos se facilitan las actividades diarias a realizar dentro de cada organización, logrando que cada proceso sea transparente y auditable.

Requerimientos de hardware y software

Para instalar y ejecutar la aplicación como servidor se requiere que el equipo posea las siguientes características:

Computador Intel Xeon 3.2 Ghz o superior.

8 GB de memoria RAM.

Monitor SVGA color.

Unidad de lectora de DVD.

Espacio libre en el disco duro: 40 GB.

Microsoft Windows Server 2008 - 2008R2, Windows Server 2012.

La licencia del producto Windows debe ser original.

Consideraciones Generales:

Debe tener instalado, como manejador de base de datos, Microsoft SQL Server 2008, 2008R2, 2012 ó 2014 en versiones Express o Standard, de acuerdo a la licencia:

- ✓ Microsoft SQL Server Express para licencias Home, Lite, Small Business, Desktop y Profesional.
- ✓ Microsoft SQL Server Standard para licencias Corporativa Small Business Y Corporativa.
- ✓ Microsoft SQL server 2014 no es compatible con los Windows menores al seven o server 2008 como: XP, Vista, server 2000 y 2003.
- ✓ Microsoft SQL Server 2008 y SQL Server 2008 R2 no son compatible con Windows 10.
- ✓ Es imprescindible tener instalado en el servidor Internet Information Server (IIS) versión 6.0. o superior.
- ✓ El Internet Information Server debe tener correctamente configurado ASP.NET 2.0. Tanto los clientes como el servidor deben tener instalado .NET Framework 3.5 (Service Pack 1).
- ✓ Es necesario tener instalado el MSI Engine 3.1 o superior.
- ✓ El rendimiento bajo estos requisitos puede variar de acuerdo a la carga de trabajo del equipo y/o la cantidad de usuarios que acceden al servidor.

- ✓ Debe asegurar la correcta configuración de su red, para permitir la comunicación vía protocolo HTTP, entre el cliente y el servidor.
- ✓ El sistema a nivel de servidor emplea servicios que se ejecutan bajo la cuenta AUTHORITY\SERVICIO la cual graba archivos temporales en carpetas dentro del equipo.

Es necesario que esta cuenta tenga asignado los permisos necesarios de escritura y lectura sobre estas carpetas donde se generaran los archivos temporales para el correcto funcionamiento del sistema.

Dichos permisos deben ser asignados por el administrador del sistema de acuerdo a las políticas de seguridad de su empresa.

A continuación se presentas dos posibles escenarios los cuales deben ser validados previamente por su administrador:

- ✓ Asignar los permisos de lectura y escritura directamente al usuario sobre las carpetas en la que se generan los temporales. Colocar en el grupo de administradores de la máquina al usuario.
- ✓ Si presenta algún problema reinicie el servicio de Internet Information Server por ventana de comandos (instrucción iisreset). En caso de Instalar sobre un Sistema operativo Windows de 64 bit debe tener instalado y actualizado correctamente el subsistema SysWOW64.

Para instalar y ejecutar la aplicación como estación de cliente, es recomendable que el equipo cuente con las siguientes características:

Las estaciones de cliente se conectan al servidor sobre el cual funciona parte de la aplicación, este servidor debe tener mínimo las siguientes características:

Computador Pentium Dual Core 3.2 Ghz

4 GB de memoria RAM.

Monitor SVGA color, resolución 1024x768 ó superior. Tarjeta de video 128 MB o superior.

Unidad de lectora de DVD.

Espacio libre en el disco duro: 4 GB.
Microsoft Windows 7 (Professional, Enterprise o Ultimate), Windows 8 (Pro, Pro WMC, Enterprise).
Windows 10 (Profesional, Enterprise).
Las estaciones de cliente se conectan al servidor sobre el cual corre parte de la aplicación, este servidor debe tener mínimo las siguientes características:
Computador Pentium Dual Core 3.2 Ghz.
4GB de memoria RAM. Monitor SVGA color.
Unidad de lectora de DVD.
Espacio libre en el disco duro: 10 GB.
Windows 7 (Professional, Enterprise, Ultimate), Windows 8 (Pro, Pro WMC, Enterprise).
Windows 10 (Profesional, Enterprise)
Windows Server 2008 – 2008 R2, Windows Server 2012.

La aplicación esta delineada siguiendo los diseños de programación que ofrecen las ultimas herramientas de Desarrollo que provee Microsoft y Visual Studio 2008:
• Team Systems.
• C # (Tecnología .net).
• SQL Server 2008 y superior
Todas estas herramientas implementadas bajo la metodología Microsoft Solutions Frameworks versión 4.0 (MSF), y empleándose la filosofía Agile Software Development.

La instalación debe ser efectuada por el usuario administrador, el cual debe tener todos los permisos necesarios (full control o administrador) para ejecutar las actividades requeridas

Ediciones de la Aplicación

La aplicación está disponible en tres (3) ediciones:

Desktop: Ideal para aquellas empresas que necesiten todo el poder y funcionalidad de la aplicación contando con un máximo un (1) usuario o sesión ejecutando el sistema a la vez. Tiene un límite de 75.000 registros contables por empresa.

Profesional: Ideal para aquellas empresas que necesiten todo el poder y funcionalidad de la aplicación. Tiene un límite de 150.000 registros contables por empresa.

Corporativa: Ideal para aquellas empresas que efectúan un número alto de transacciones y necesitan todo el poder y funcionalidad del sistema, sin límites en el número de usuarios o sesiones que van a ejecutar el sistema a la vez.

Todas las ediciones requieren la adquisición por parte del cliente del manejador de base de datos SQL Server de Microsoft 2008 o superior.

Dichas ediciones funcionan bajo el paradigma Cliente/Servidor, lo que le permite manejar mayores volúmenes de datos con tiempos de respuesta adecuados y una excelente confiabilidad.

Implantación de la Aplicación

La implantación o puesta en funcionamiento del sistema en su empresa, más que un proyecto de informática, debe ser vista como un proyecto de negocios que involucra a los usuarios de cada uno de los procesos que se ejecutan en la empresa.

Para implantar el sistema es necesario ejecutar un conjunto de actividades, algunas de las cuales son de tipo técnico, pero la mayoría son de tipo funcional.

Las actividades más importantes a llevar a cabo son:

- Instalación del sistema.
- Análisis de los procesos actuales del negocio.
- Parametrización del sistema.
- Carga de datos.
- Diseño de los formatos de la empresa.

- Adiestramiento de los usuarios.
- Pruebas y validación del funcionamiento del sistema.

.

Es importante tener claro que al implantar el sistema, los requerimientos de la empresa deben ser satisfechos mediante la configuración o parametrización del sistema.

La aplicación cuenta con las siguientes características generales:

Interfaz gráfica: El sistema fue desarrollado utilizando las herramientas de programación contenidas en Visual Studio 2008 de Microsoft, respetando al máximo la filosofía de diseño y las normas de operación de las aplicaciones Windows. Su interfaz gráfica, cómoda e intuitiva sigue el paradigma de cinta o Ribbon empleado por la nueva generación de aplicaciones de escritorio de Microsoft. Dicho paradigma tiene como principal característica la organización de las opciones bajo una óptica funcional, a fin de permitir al usuario un rápido acceso a ellas.

Multi - empresa: El sistema permite la creación y manejo de tantas empresas como desee. Cada empresa puede ser configurada en función de sus características y su creación puede ser efectuada partiendo desde cero o de otra Empresa que ya exista

Flexible y altamente parametrizable: Gracias a su alta flexibilidad la aplicación se adapta perfectamente al funcionamiento de empresas de diversos tamaños en sectores tan diferentes como comercio al mayor, detal, fábricas y empresas de servicios. Su gran flexibilidad lo convierte en una poderosa herramienta para la gestión de su empresa.

Seguro y auditable: La aplicación permite crear mapas de usuarios en los que se define en forma rápida y sencilla las tablas, procesos, y grupo de reportes que van a poder utilizar los usuarios asociados a cada mapa.

Adicionalmente el sistema le permite seguir las “pistas” de las operaciones efectuadas por los usuarios, para las tablas y procesos del sistema; pues guarda en los principales registros la siguiente información:

- Usuario que realizó la operación.
- Fecha y hora en la que se efectuó la operación.
- Operación: I=Ingresar registro, M=Modificar registro, E=Eliminar registro.
- Datos modificados: valor anterior y valor nuevo.
- Nombre del equipo donde se hizo la operación.

Despliegue de reportes: Usted podrá desplegar los resultados de sus reportes de forma dinámica, mediante el uso de un explorador en profundidad *(dril down*), que le permitirá aislar partes de los resultados (por ejemplo una jerarquía de cuentas) en un sub-reporte. Adicionalmente, cuenta con la opción de búsqueda de texto así como la exportación de los resultados a los principales formatos electrónicos.

Generador de formatos y reportes: Tiene incorporado más de cien (100) formatos y reportes, los cuales pueden ser modificados haciendo uso del generador de informes *Crystal Reports*, ubicado en la Herramienta de Administración

.

Además nuevos reportes y formatos pueden ser agregados al sistema haciendo uso de dicha funcionalidad.

Adaptado a la normativa vigente: La aplicación fue desarrollada en Venezuela por un grupo de profesionales altamente competentes en desarrollo del software, razón por la cual se adapta totalmente a la normativa vigente en Venezuela, en materia contable e impuesto sobre la renta.

Diseño Multicapas: La aplicación tiene un avanzado diseño en multicapas que le permite manejar grandes volúmenes de información en forma eficiente y confiable.

La aplicación, en cualquiera de sus ediciones, es compatible con los sistemas operativos de red más importantes de Microsoft ®

.

Herramienta de Administración: Las opciones de mantenimiento y administración del sistema son manejadas desde un programa adicional, el cual centraliza las actividades del administrador del sistema en un solo lugar e independientemente de las actividades funcionales del día a día.

Seguridad de la aplicación

- ✓ Incorporación de las tablas de Usuarios, Mapas de Acceso en la Base de Datos de la empresa.
- ✓ Manejo de Status en los usuarios: Activo, Inactivo, Bloqueado (bloqueo por intrusos), Contraseña expirada.
- ✓ Posibilidad de conformar políticas de vencimiento de contraseñas por caducidad y/o inactividad, bloquear los usuarios después de una cantidad configurable de intentos fallidos de entrada al sistema y definir los minutos de duración del bloqueo.
- ✓ Posibilidad de autenticar a los usuarios a través del Active Directory.

Tecnología a sus Alcance

La aplicación ofrece los beneficios de la integración de SQL y las ventajas de la tecnología cliente / servidor las cuales son bien conocidas por las grandes empresas, razón por la cual la mayoría de los sistemas enfocados a este mercado hacen uso de la tecnología SQL como manejador de base de datos.

Con la aplicación se obtiene un sistema de información para toda la compañía, que proporciona características tan avanzadas como: Procesamiento de transacciones a alta velocidad e integridad de datos, Seguridad definida por el usuario, Escalabilidad del Sistema y Flexibilidad Modular.

Sus herramientas sólidas y lenguaje centrado en datos y orientado a objetos lo convierte en el software ideal para crear aplicaciones Multi niveles, escalables, y modernas que integren la computación cliente / servidor y el Internet.

Dentro de los servicios de post venta brindados se encuentran los siguientes:

Soporte Remoto: la sólida tecnología utilizada lo apoyara, a la brevedad posible, a través de soporte remoto. Este servicio permite brindar ayuda en la operación del sistema, sin tener que esperar la visita del técnico a la empresa, solo con la conexión de internet y la segura tecnología de acceso y control remoto, se obtendrá rapidez, efectividad y economía, tres ventajas alineada a los intereses.

Soporte en Sitio: en ocasiones no es posible ejecutar el soporte remoto, por lo cual se brinda el soporte en el sitio con la finalidad de garantizar el funcionamiento y operatividad del sistema.

Desarrollo de Producto Software a la Medida

El producto de software es adaptado a la medida de las necesidades de la empresa, del negocio, permite gestionar de forma optimizada una empresa, y los procesos de gestión que en ella se llevan a cabo: ejemplo pedidos, facturas, gestión de rutas de reparto, etc. A través del software a la medida se pueden reducir gastos (ejemplo: optimizar trámites, minimizar tiempos de entrega de un pedido, gestionar rutas óptimas de entrega de pedidos) y aumentar las ventas (ejemplo: tienda online, difusión en redes sociales, etc.).

Para adaptar el software a las funcionalidades del negocio, es preciso estudiar los siguientes aspectos:

- ✓ Realizar un estudio de factibilidad
- ✓ Generar un levantamiento de información.
- ✓ Elaborar un informe con el análisis y diseño del requerimiento especial para cada cliente.

Una vez aprobado el requerimiento, se genera el proyecto, se envía al cliente la planificación, luego se programa y se realizan las pruebas para enviar al cliente la solución.

Finalmente se da seguimiento al cliente; ya sea vía telefónica, a través de un correo electrónico o visitas programadas donde el cliente indica, manifiesta que el requerimiento satisface todas sus necesidades, de ser así se procede a cerrar con éxito el proyecto.

Uso de Garantía del Producto del Software

Hacer un uso lícito del software no sólo proporciona garantías al usuario, sino que permite que aumenten las oportunidades de competitividad, productividad y mejora de la industria, redundando en el beneficio del usuario, ya que ayuda a los fabricantes a seguir investigando e innovando en cuanto a mejora técnica del producto; un desarrollo del sector que llevará al usuario a acceder a la última tecnología en mejores condiciones y con la mayor optimización y garantía. En el anexo a, se visualiza el contrato de licencia y uso de garantía de programas contrato de plan de asistencia y servicio.

5.2 Objetivo Nro. 2: Analizar las mejores prácticas en el área de gestión de pruebas para proyectos de Software.

Toda organización desear prosperar y alcanzar el éxito dentro de su área profesional. Para ello, debe establecer una serie de metas u objetivos de negocio a alcanzar, es decir, definir claramente qué es lo que la empresa debe conseguir para mejorar y garantizar que los procesos organizativos conducen a la empresa hacia el éxito buscado.

Se observa un modelo de cómo las etapas de pruebas se integran en el ciclo de vida de desarrollo de software genérico. Durante la etapa de planificación es importante establecer una buena estrategia de pruebas y seleccionar las técnicas adecuadas de estimación en función de los factores que afecten a las pruebas del proyecto. La siguiente fase de desarrollo es el diseño del producto, que trae consigo el plan de casos de prueba. Durante las siguientes fases de codificación,

se ejecutan las pruebas del producto, unitarias, de sistemas, de integración, etc., de las que se explicará en los apartados siguientes.

El proceso de validación puede considerarse como un subproyecto dentro del procedimiento sobre el cual se están ejecutando las pruebas, y como tal requiere la definición de un plan a seguir. Cuando el proceso de validaciones existe dentro del contexto del proyecto, debería prestarse atención a la efectividad y eficiencia de las aprobaciones desde la perspectiva del proyecto y no desde la perspectiva del propio subproyecto de pruebas.

La eficiencia consiste en conseguir el efecto deseado de la manera correcta, es decir, sin desaprovechamiento de recursos, ni de tiempo ni de dinero. Por consiguiente, la eficiencia está relacionada con dos conceptos: productividad y ausencia de pérdidas. Para conseguir esta eficiencia deseada durante el proceso de pruebas, se pueden considerar los siguientes aspectos:

- ✓ **Evitar redundancias:** Las redundancias traen consigo una pérdida o desaprovechamiento de tiempo por lo que hay que intentar ejecutar las pruebas más adecuadas a cada situación y lo más rápido posible. Es importante encontrar los errores que más impacto puedan tener sobre el producto lo más rápido posible. Aunque sea aconsejable no desaprovechar el tiempo, no hay que olvidarse de la planificación, preparación de las pruebas, y de prestar atención a todos los detalles. No abandonar las estrategias previamente establecidas ante la primera señal de problemas o retrasos. Es decir, en un intento de ahorrar tiempo, se debe tener cuidado de no cometer errores que tendrán como consecuencias invertir más tiempo del que se intenta ahorrar.

- ✓ **Reducir costos:** Para reducir los costos se debería prestar especial atención a la adquisición de elementos que puedan ayudar a la ejecución de validaciones, del tipo de herramientas para la ejecución de pruebas o

entornos de las mismas. Habría que cerciorarse de que realmente fueran necesarias y de que existe el tiempo y las habilidades suficientes para emplearlas de manera ventajosa. También, es aconsejable evaluar las herramientas antes de comprarlas y ser capaz de justificar estos gastos frente al análisis de costos-beneficios.

Según la (IEEE, 1990), un caso de prueba es un conjunto formado por entrada de datos, condiciones de ejecución y resultados esperados, desarrollados para un objetivo particular. En el caso concreto de un caso de prueba funcional del sistema; esta información se obtendrá de las especificaciones funcionales del sistema. Así un caso de prueba, será un fragmento del comportamiento descrito en un requisito funcional perteneciente a las especificaciones del sistema. Durante la ejecución de un caso de validaciones funcionales, será necesario contar con un elemento que interactué con el sistema mediante la interfaz adecuada para realizar el requisito funcional. La misión de este elemento será realizar los pasos de casos de prueba, verificando que el sistema exhibe las respuestas esperadas.

Validación y Verificación en el desarrollo de software

Los procesos de validación y verificación determinan si un aplicativo satisface las necesidades del negocio y si se está construyendo acorde a las especificaciones. La certificación del software es la evaluación del grado de cumplimiento de los requisitos establecidos para el producto del software.

Resulta evidente que para poder validar el grado de corrección de un producto software es necesario saber qué es lo que se espera de él. El comportamiento correcto puede expresarse a través de la especificación del software, la misma puede hacerse de manera informal o siguiendo una metodología. El principal problema de la especificación informal es la ambigüedad, ya que muchas veces las mismas pueden tener diversas interpretaciones. Otro problema añadido a está en que si no se revisa adecuadamente la documentación, la especificación puede que sólo sea comprensible por el propio desarrollador del software.

Con respecto a las tareas asociadas a estos procesos, las pruebas están más relacionadas con el paso a paso de validación, mientras que las revisiones son tareas más orientadas a las pruebas de verificación.

El objetivo principal de la validación y verificación es comprobar que el sistema está hecho para un propósito, es decir, que el sistema debe ser lo suficientemente bueno para su uso previsto, cumpliendo con las especificaciones del alcance. El nivel de confianza requerido depende de tres factores:

- El propósito o función del sistema. El nivel de confianza necesario depende de lo crítico que sea el software para una organización.
- Las expectativas del usuario. Actualmente, es menos aceptable entregar sistemas no fiables, por lo que las compañías deben invertir más esfuerzo en llevar a cabo las actividades de verificación y validación.
- Entorno de mercado actual. Cuando un sistema se comercializa, los vendedores del mismo deben tener en cuenta los sistemas competidores, el precio que sus clientes están dispuestos a pagar por el aplicativo y los plazos requeridos para entregar dicho software. Cuando una compañía tiene pocos competidores, puede decidir entregar un programa antes de que haya sido completamente probado y depurado, debido a que quiere ser el primero en el mercado. Cuando los clientes no están dispuestos a pagar precios altos por el software, pueden estar dispuestos a tolerar más defectos en él.

Tipos de pruebas

La disciplina de pruebas es una de las más costosas del ciclo de vida software. En sentido estricto, deben realizarse las pruebas de todos los artefactos generados durante la construcción de un producto, lo que incluye especificaciones de requisitos, casos de uso, diagramas de diversos tipos y, por supuesto, el código fuente y el resto de productos que forman parte de la aplicación (por ejemplo, la base de datos), e infraestructura. Obviamente, se aplican diferentes técnicas de prueba a cada tipo de software.

A continuación, se describirá los tipos de pruebas en función de qué conocemos, según el grado de automatización y en función de qué se prueba.

a. Pruebas de caja negra

En este tipo de prueba, tan sólo, se puede probar dando distintos valores a las entradas. Los datos de prueba se escogerán atendiendo a las especificaciones del problema, sin importar los detalles internos del programa, a fin de verificar que el programa funcione bien.

Este tipo de validaciones se centra en los requisitos funcionales del software y permite obtener entradas que ejecuten todos los flujos de una funcionalidad (casos de uso).

Con este tipo de pruebas se intenta encontrar:

- Funcionalidades incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y finalización.

b. Pruebas de caja blanca

Consiste en realizar pruebas para verificar que líneas específicas de código funcionan tal como está definido. También se le conoce como prueba de caja transparente, el cual es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de las mismas.

Las pruebas de caja blanca intentan garantizar que:

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se ejecuten todos los bucles en sus límites
- Se utilizan todas las estructuras de datos internas.

- Para esta prueba, se consideran tres importantes puntos. Conocer el desarrollo interno del programa, determinante en el análisis de coherencia y consistencia del código.

c. Pruebas de Estrés

El objetivo de esta prueba es testear el funcionamiento del sistema bajo condiciones extremas de demandas de recursos.

Las pruebas de estrés se deben ejecutar a la aplicación de manera tal que demande recursos en cantidad, frecuencia, volúmenes anormales. Por ejemplo: pruebas que generen una gran cantidad de transacciones por segundo, que incrementen en un orden de magnitud los volúmenes de datos almacenados para comprobar los tiempos de respuesta, ejecutar casos de prueba que requieren el máximo de memoria o de otros recursos.

Herramientas para ejecución y registro de pruebas

En este apartado, se van a definir tres tipos de herramientas relacionadas con la ejecución y registro de pruebas las cuales se detallan a continuación:

- ✓ **Herramientas de ejecución de pruebas**

La mayoría de las herramientas de este tipo ofrecen un mecanismo para la captura y registro de pruebas. Las mismas suelen utilizar un lenguaje de *scripting*, por lo que si los analistas de pruebas desean utilizar una herramienta de ejecución de pruebas necesitarán tener habilidades de programación sobre la creación y modificación de *scripts*.

- ✓ **Comparadores de pruebas**

La esencia de las pruebas es comprobar si el software produce el resultado correcto, y para ello deben comparar que el mismo produce los resultados que se esperan. Los comparadores de pruebas ayudan a automatizar estos aspectos. Hay dos maneras de llevar a cabo esta comparación. De manera dinámica, donde la misma es realizada activamente,

es decir, mientras la prueba se está ejecutando, o post ejecución, donde la comparación se realiza después de que la prueba haya acabado y el software bajo estas condiciones no se vuelva a ejecutar.

- ✓ **Herramientas de seguridad**

En la actualidad existen muchas herramientas de seguridad diseñadas específicamente para analizar la confiabilidad de un sistema y para ofrecer cierto grado de seguridad a una red.

Las herramientas de pruebas de seguridad se utilizan para probar la seguridad que tienen los sistemas, intentado acceder a un sistema, por ejemplo, también, se encargan de identificar virus, simular ataques externos, detectar intrusos, etc.

Procedimiento de Gestión de Pruebas

Planificación de Proyectos de Software

La Planificación es un proceso que comienza con una misión, metas y objetivos que deben lograrse. Desarrolla planes, procedimientos, establece una organización y asigna recursos y responsabilidades con el propósito de alcanzar los objetivos propuestos. El resultado principal de la planificación es el Plan del Proyecto.

Objetivos de la Planificación de Proyectos de Software

El principal objetivo de la planificación en proyectos de desarrollo de software es ordenar el qué hacer durante el mismo y asignar adecuadamente los recursos y tareas para cumplir los objetivos propuestos. En general se planifica para:

- ✓ Organizar el qué hacer del proceso de desarrollo de software.
- ✓ Minimizar tiempo y costos involucrados.
- ✓ Maximizar el uso de recursos disponibles.
- ✓ Establecer hitos del proyecto.

- ✓ Medir el avance.
- ✓ Mejorar la comunicación.
- ✓ Obtener soporte técnico, de gerencia y político.

La planificación es una tarea que se desarrolla al inicio del proyecto pero administra el resto de las fases. Una buena proyección inicial ayudará a que las metas propuestas se cumplan y que los eventuales inconvenientes sean abordados de mejor forma.

El proceso de planificación produce idealmente un conjunto de planes, clasificados como esenciales y de soporte.

Los planes esenciales son aquellos que se consideran imprescindibles en cada proyecto, dentro de estos están: Plan de Proyecto, Plan de Pruebas y Plan de Instalación. Los planes de soporte no siempre son necesarios, entre ellos están: Plan de Entrenamiento, Plan de Control de Cambios.

Plan de Pruebas

El plan es el medio de comunicación en el proceso de *testing*. Se inicia después de los requerimientos y su prerrequisito es que exista un estándar de pruebas. Un buen plan debe permitir planificar, monitorear y controlar todas las actividades del mismo.

Objetivo del Plan

Debe contener niveles de prueba: módulo, integración, sistema. Tipos de prueba: funciones, rendimiento, documentación. Aspectos que no se probarán como parte de este plan: instalación, características especiales.

Misión de las Pruebas aplicable a este proyecto

La misión de la evaluación para el proyecto de gestión de pruebas de software se define enfocada al aseguramiento de la calidad de los componentes y módulos tecnológicos desarrollados, de manera que estos cumplan con la especificación de los requerimientos del cliente. Para esto se definen los siguientes lineamientos que constituyen la misión y objetivos dentro este proceso:

- Descubrir tantos errores como sea posible
- Notificar acerca de los riesgos percibidos del proyecto
- Examinar la aplicación para comprobar si hace o no lo que se supone, debe hacer. De igual forma verificar si ésta hace o no lo que se supone, no debe hacer.
- Validar y Verificar a través de la comparación del resultado de las pruebas del aplicativo con el resultado que el mismo tendría que producir de acuerdo a su especificación.
- Evaluar la calidad del producto y satisfacción de los interesados
- Cumplir con los requerimientos del cliente

El proceso de evaluación y pruebas debe permitir detectar problemas desde el inicio de la especificación de requerimientos, antes de que sean de gran impacto en fases más adelantadas del proyecto, esto con el fin de disminuir los riesgos y de obtener un producto con calidad logrando mayor satisfacción del cliente

Roles y Responsabilidades

La definición de roles se desarrolló teniendo en cuenta la estructura organizativa del departamento de la Gerencia de Tecnología de la Información GTI, así como también la preparación técnica del equipo del proyecto. Cada rol viene definido con sus responsabilidades.

Riesgos asociados al desarrollo de pruebas del software

En esta sección se identifican y analizan los riesgos del proyecto a fin de planificar la respuesta a los mismos y en plan de acción a ejecutar. Para llevar a cabo la planificación de los riesgos se tomaron las siguientes acciones:

- ✓ Revisión de los procesos presentes para el plan de gestión de pruebas.

- ✓ Reunión entre el líder de tecnología, líder de pruebas y el gerente de proyectos para hacer una lluvia de ideas donde se identifiquen los riesgos potenciales y sus posibles respuestas.
- ✓ Revisión de lecciones aprendidas y riesgos documentados en un proyecto similar, donde se realicen y ejecuten pruebas de software.

Un riesgo es un evento o condición incierta que, si sucede, tiene un efecto en por los menos uno de los objetivos del proyecto, puede traer una o varias causas y si sucede uno o más impactos.

Una vez analizados los mismos que son potencialmente más peligrosos, es importante recurrir a distintas técnicas de control de los mismos. Por ejemplo, se pueden elaborar planes de contingencia para los riesgos que sean más probables y de consecuencias más desastrosas para el proyecto

Métricas relacionadas con el proceso

Las revisiones técnicas son una de las muchas acciones que se requieren como parte de las buenas prácticas de la ingeniería de software. Cada acción requiere un esfuerzo humano dirigido. Como el esfuerzo disponible para el proyecto es finito, es importante que una organización de software comprenda la eficacia de cada acción, definiendo un conjunto de métricas.

Las mejoras en el proceso de desarrollo de software y sistemas de calidad no pueden ser evaluadas sin un esfuerzo efectivo de medición. Cada organización desea mejorar sus procesos en la programación del software, debido a que existe un tangible beneficio con la construcción de un mejor software.

A continuación se enumeran las siguientes necesidades de medición:

- ✓ Mejoras en la calidad y productividad.
- ✓ Planificación y estimación de proyectos con alguna precisión.
- ✓ Disposición del personal adecuado, bien utilizado y motivado.
- ✓ Existencia de una adecuada estructura organizacional.
- ✓ Uso de técnicas y herramientas efectivas para el proceso.
- ✓ Obtención de un espacio físico y ambiente de trabajo óptimo.

Las métricas relacionadas con el proceso de gestión de pruebas del software se pueden diseñar para proporcionar información específica sobre el producto a desarrollar.

Uno de los objetivos del seguimiento del proceso es descubrir oportunidades para la mejora del proceso. Las métricas del proceso proporcionan un buen medio para monitorizar la efectividad de las actividades de las pruebas del desarrollo del software. El análisis de estas métricas conducirá a los correspondientes cambios en el proceso. Una vez recabadas las métricas, se usan para evaluar la eficacia de las revisiones que se efectúen.

Por ejemplo, para evaluar los criterios que determinan que niveles de aprobación son óptimos para ciertos tipos de cambios sería útil la información acerca del tiempo requerido para abordar dichos cambios.

Algunos ejemplos de indicadores que se pueden tomar del proceso son:

- ✓ Número de no conformidades relativas a la integridad de los productos desarrollados
- ✓ Número de cambios y su estado.

Medición de la Extensión de las Pruebas

Cuando se tiene un número determinado de casos de prueba por cada uno de ellos, la forma de medir la extensión de las pruebas será comparando el número de casos de prueba ejecutados satisfactoriamente contra el número de prueba total, esto nos dará a conocer el porcentaje de pruebas ejecutadas por el grupo de *testing*.

Extensión de las pruebas = (Casos de prueba ejecutados satisfactoriamente *100)/total de casos de prueba

Datos de Prueba

Con el objetivo de realizar unas pruebas acertadas y lo más cercanas a la realidad que sea posible, es necesario contar con datos que alimenten la ejecución de los casos de prueba, los cuales, en la medida de lo posible, deben ser reales, cubrir un rango considerable y representar una profundidad significativa dentro del

dominio de los datos que maneja la organización en los procesos de negocio involucrados.

El set de datos de prueba debe cumplir con la estructura del modelo de datos del negocio, y debe ser generados como una base de datos relacional que respete la integridad referencial requerida por el proceso (relaciones, jerarquía, restricciones etc.).

Políticas de Administración de los Datos de Prueba

Una vez construido el set de datos de prueba, el mismo es administrado por el equipo de proyecto siguiendo las políticas expuestas a continuación:

- Se debe realizar un *backup* inicial del set de datos, antes de cualquier otro tratamiento, y este debe ser etiquetado apropiadamente para su posterior identificación entre los demás respaldos que se llevarán a cabo.
- Se deben hacer *scripts* SQL que garanticen la integridad referencial del set de datos de prueba construida, de cara al modelo de datos que soporta la aplicación. El set de datos solo será aceptado si la ejecución de dichos scripts de verificación no arroja inconsistencias en las relaciones de los datos.
- El set de datos se implanta en el ambiente de pruebas
- La administración del set de datos de prueba queda únicamente asignada a los responsables fijados por el equipo de desarrollo para tal fin, se debe garantizar el acceso a los mismos y a los *backups* haciendo uso de la seguridad que dispone el motor de base de datos utilizado
- Los respaldos del set se realizan de la siguiente manera de acuerdo al ambiente como se muestra en la Tabla N° 8 :

Tabla N° 8. Administración de los Datos de Prueba.

Ambiente	Política de *BackUps*
Ambiente de Pruebas	• Un *backup* completo antes de la ejecución de cualquier caso de prueba. • Al finalizar una jornada de pruebas se realiza un *backup* incremental. • Al detectar errores de impacto "Crítico" en la aplicación, se realizará *backup* completo e inmediato de la base de datos y del sitio de la aplicación, asociándolo a un documento descriptivo del escenario y del caso de prueba que se estaba llevando a cabo. Esto con el fin de poder reproducir estos errores posteriormente sobre un ambiente controlado y facilite la detección de la causa y la corrección del mismo.

- La restauración del set de datos de prueba a su estado inicial sobre un ambiente se dará cuando ocurra alguno de los siguientes eventos:
  - ✓ Se detecta corrupción de los datos (perdida de integridad referencial, errores de metadata, referencia a valores no existentes etc.) por causa de errores en la aplicación o por el manejo inadecuado de los datos por parte de los desarrolladores de a nivel del motor de base de datos
  - ✓ Se libera una nueva versión del set de datos de prueba.
  - ✓ Se detectan errores en la aplicación causados por integridad referencial en los datos que no fueron advertidos en los controles iniciales.
  - ✓ Se libera un nuevo *release* de la algún componente que implica pruebas de regresión y de nuevas funcionalidades

Ciclo de Vida de las Pruebas

La ejecución de pruebas de un sistema involucra las etapas que a continuación se detallan:

Planificación de pruebas.
Diseño y construcción de los Casos de Prueba.
Preparación del ambiente de pruebas y generación de datos de prueba.
Ejecución de las pruebas.
Registro de errores encontrados.
Registración de resultados obtenidos.

Depuración de los errores.
Informe de los resultados obtenidos

En el marco del proceso definido para el área, las etapas mencionadas forman el ciclo de vida de las pruebas el cual se integró al proceso de desarrollo de la aplicación de la siguiente manera:

- ✓ Durante la fase de elaboración se defina el plan de pruebas.
- ✓ En la fase de elaboración se definan los casos de prueba.
- ✓ Para la fase de construcción, se diseñan los casos de prueba y se generan las bases de datos de prueba.

Posteriormente se ejecuten todos los *scripts* y programas de pruebas anteriormente desarrollados y se realice un adecuado seguimiento y control.

Proceso de Prueba

El proceso de prueba se compone de los siguientes pasos:

1. El circuito comienza con la generación del plan de pruebas por parte del líder del proyecto en la fase de elaboración. El mismo se generará una vez aprobado el plan de proyecto para el área al cual pertenece el sistema a probar.
2. Luego en la fase de construcción, el analista de pruebas del proyecto diseñará las pruebas basándose en la documentación y conocimiento del software a evaluar, esto dará como resultado tantos documentos de casos de prueba como *testers* sean designados.

Esta actividad dependerá de la situación:

- En la prueba del sistema o aplicación existirá un documento de casos de prueba por *tester*, en éste el analista de pruebas completará el ambiente de prueba a utilizar, los cuáles serán los datos de entrada, con los resultados esperados.
- Si se está probando un sistema por primera vez, a parte de los documentos de casos de prueba que se generen por *tester*, se deberá crear un documento general de casos de verificación que reúna todos los eventos de las mismas, para reutilizar en futuras pruebas.

- Si se está evaluando una nueva operación, se adicionaran tantos casos de prueba como sean necesarios al final del documento general, donde se especificará la situación por la cual se generaron dichos casos.
- Si se está probando (un *script* ya sea de consulta, actualización, procedimiento almacenado, etc.) se deberá generar un documento de caso de prueba que contemple la funcionalidad e impacto del mismo en el sistema.

3. Una vez que el documento de casos de prueba fue aprobado, inicialmente por el líder del proyecto, éste pasará al *tester*.

4. El *tester* previamente verificará que el ambiente de prueba cumple con los requisitos y ejecutará los casos con los datos de validación proporcionados, documentando los resultados obtenidos y conclusiones inferidas en el mismo documento.

5. A partir de los resultados obtenidos, y en el caso de que existieran *bug* el analista de sistemas compilará los errores registrados en los distintos documentos de casos de prueba en un único reporte de errores, en éste se dejará constancia de todos las fallas, errores detectados, este documento es de características técnicas, dirigido principalmente al implementador con el objetivo de que se corrijan las fallas. El reporte de errores estará asociado a varios documentos de casos de prueba, será actualizado por el implementador en los casos en que corresponda.

6. En los sistemas de código cerrado el reporte de errores también será generado por el analista de sistemas con el objetivo de enviar a la entidad que corresponda y si existiera una solución proporcionada por los mismos, será registrada por él en el mismo documento.

7. En los sistemas o aplicaciones de código abierto, a partir de la detección se realizará la depuración (localización y corrección de defectos si correspondiere).

8. La depuración estará a cargo del implementador, el cual puede o no corregir los errores. Es su responsabilidad completar la acción tomada en el documento reporte de errores, columna acción tomada.

9. Si se corrige un error, se debe volver a probar el software para comprobar que el problema este resuelto y cumple con la funcionalidad requerida, en este caso el reporte de errores completo y el producto a evaluar volverá al analista de sistemas quien definirá un nuevo documento de casos de prueba, en el que se menciona si se trata de una ejecución o pre ejecución, este vuelve al *tester* quien en el caso en que persistan los mismos errores o nuevos, los registrará en el mismo reporte de errores inicial, a partir de acá el proceso de repite para todas las situaciones en las que se continúen encontrando errores.
10. Una vez finalizadas las pruebas y depurados los errores encontrados el líder del proyecto generará el informe de resultados obtenidos el cual va dirigido, a la Gerencia TI, según corresponda, el mismo contiene un resumen de los resultados de las pruebas ejecutadas, aporta una evaluación del software basada en dichos resultados y una descripción del análisis de errores encontrados el cual servirá para realizar predicciones de la fiabilidad del mismo, detectar las causas más habituales de error y mejorar los procesos de desarrollo y prueba.
11. Una vez revisado volverá al líder de proyecto para que este lo almacene en el directorio correspondiente del espacio de trabajo.

Metodología de las Pruebas

Tipos de Prueba

Si bien el área de estudio de las pruebas es muy amplia, a continuación de describen un conjunto de los tipos de pruebas que pueden ejecutarse en cada proyecto, las mismas se clasifican en las siguientes categorías:

Pruebas Cajas Blanca

Las pruebas de caja blanca normalmente se denominan pruebas de cobertura o de caja transparente, al total de pruebas de caja blanca se le llama cobertura, la misma es un número porcentual que indica cuanto código del programa se ha probado.

Básicamente la idea de pruebas de cobertura consiste en diseñar un conjunto de validaciones en las que se va ejecutando sistemáticamente el código hasta que se haya ejecutado todo o la gran mayoría de él, si bien en algunos casos puede ser complicado una vez realizada se tiene la seguridad del funcionamiento de todos los módulos.

Las pruebas de caja blanca no reemplazan, solo complementan a las de caja negra y de aceptación, deben ser realizadas una vez terminado el software y no deben ser confundidas con las pruebas informales que realiza el programador durante el desarrollo, dado que si bien estas van cubriendo distintos fragmentos de código, no tienen un diseño apropiado.

En el plan de pruebas del software, se especificó si se utilizó este tipo de pruebas, en que módulos y la forma en que se llevaron a cabo.

Pruebas Cajas Negras

Son pruebas funcionales, de entrada/salida o inducidas por los datos.

Las pruebas de caja negra se centran en lo que se espera de un módulo, intentan encontrar casos en que el módulo no se ajusta a su especificación. Por ello se denominan pruebas funcionales, y el *tester* se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario. Las mismas se apoyan en la especificación de requisitos del módulo. Se utiliza el término "cobertura de especificación" para dar una medida del número de requisitos que se han probado, lo más recomendable será conseguir una alta cobertura en esta línea.

Estrategia de Prueba

Las estrategias de software intentan agrupar diversos tipos y técnicas de prueba, a fin de asegurar que se hayan probado todos los niveles del software.

Se utilizaron los siguientes niveles para clasificar las pruebas.

Nivel de Unidad

En este nivel se prueba cada unidad o módulo con el objetivo de eliminar errores en la interfaz y en lógica interna. Esta actividad utiliza caja negra y blanca, según convenga para lo que se desea probar.

Se deben diseñar pruebas para descubrir los siguientes errores:

- Tipificación impropia o inconsistente.
- Inicialización o valores implícitos erróneos.
- Nombres de variables incorrectos.
- Tipos de datos inconsistentes.
- Errores de cálculos: Dentro de este tipo de error se encuentran los siguientes:
  - ✓ Precedencia aritmética incorrecta o mal interpretada.
  - ✓ Inicializaciones incorrectas.
  - ✓ Falta de precisión.
  - ✓ Incorrecta representación simbólica de una expresión.
- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o de precedencia incorrectos.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.
- Variable o comparaciones incorrectas.
- Terminación de bucles inapropiada o inexistente.
- Fallo de salida cuando se encuentra una iteración divergente.
- Bucles que manejan variables modificadas en forma inapropiada.

Nivel de Integración

El objetivo de estas pruebas es tomar los módulos auditados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

En este tipo de prueba se observa cómo se acoplan los distintos módulos, ya que lo que resulta de un módulo puede no ser lo que espera otro. Para esto existen dos enfoques fundamentales: uno ascendentes y otro descendente.

- ✓ Ascendentes: empieza agrupar desde los módulos de mayor jerarquía, avanzando progresivamente hacia los que poseen menor jerarquía.
- ✓ Descendente: agrupa los módulos de menor jerarquía en racimos y avanza hacia los de menor jerarquía.

Independientemente del enfoque que se utilice, lo más importante es realizar la integración de manera ordenada y, en caso de encontrar un error corregirlo y aplicar el concepto de regresión.

Este concepto expresa que ante la aparición de un error, se debe solucionar y luego realizar al módulo una nueva prueba de unidad.

Nivel de Sistema

Tras la culminación de las pruebas de integración, el sistema estará completamente ensamblado y ya se corrigieron los errores de las pruebas unitarias y de integración. El objetivo de las mismas es verificar que el sistema cumpla con los requerimientos funcionales y no funcionales definidos por el usuario, se deben observar aspectos como la recuperación del aplicativo, la seguridad, la resistencia, la escalabilidad y el rendimiento estándar.

Requerimientos Funcionales

La validación del software se consigue mediante una seria de pruebas de caja negra que demuestran conformidad con los requerimientos. Estas validaciones pueden ser ejecutadas por el usuario apoyado y dirigido por el equipo de pruebas.

Las técnicas a utilizar son las siguientes:

- **Pruebas Alfa**

  Son conducidas por el usuario en el ambiente de prueba preparado para tal fin. El usuario utiliza el sistema en forma natural bajo la vigilancia del *tester* quien ira registrando de los errores y problemas encontrados.

- **Pruebas Beta**

  Son realizadas por distintos usuarios finales en su lugar de trabajo. La prueba Beta es una ejecución del sistema en un ambiente controlado, aunque sin la presencia del equipo de prueba. El usuario registra todos los problemas detectados, reales y/o imaginarios y los informa al equipo de prueba en los intervalos definidos en el Plan de Prueba.

Requerimientos No Funcionales

Teniendo en cuenta el nivel de detalle técnico de estas pruebas no se requiere la participación del usuario durante la ejecución de las mismas.
Las pruebas a realizar son las siguientes:

- **Prueba de Recuperación**

Los sistemas a probar deben tener la capacidad de recuperarse ante cualquier contingencia, de acuerdo con los requerimientos no funcionales fijados por el usuario. En este tipo de pruebas se fuerza el fallo del software de muchas formas y se verifica que la recuperación se lleva a cabo apropiadamente.

Si la recuperación es automática hay que evaluar la correcta reinicialización de:

- Los mecanismos de recuperación del estado del sistema.
- La recuperación de datos.
- El proceso de re-arranque del sistema.
- Si la recuperación de la falla requiere de la intervención humana, hay que evaluar los tiempos de corrección para determinar si están dentro de los límites especificados.

- **Pruebas de Seguridad y Control de Acceso**

La prueba de seguridad debe asegurar razonablemente el cumplimiento de los requerimientos no funcionales relacionados con los objetivos de confidencialidad, disponibilidad, integridad y exactitud de la información.

Con estas pruebas se verificarán los mecanismos de protección incorporados en el sistema que lo protegerán de accesos ilegales
En cuanto a los mecanismos propios de la aplicación deberá constatarse que

- Las contraseñas no sean visibles al ser ingresadas.
- Se encuentren encriptados y almacenadas fuera del acceso público.
- Caducidad de sesiones de usuarios
  Debe verificarse además que el esquema de copias de respaldo sea el adecuado. Se debe prever, en el marco de un simulacro de contingencia, la restauración de los datos respaldados y como registra el sistema el log de transacciones

- **Pruebas de Performance**

Esta prueba está diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

La validez de estas pruebas está sujeta a poder simular todas las condiciones de funcionamiento del ambiente de producción, entre otros factores plataforma de hardware, líneas de comunicaciones y concurrencia de usuarios.

- **Pruebas de Volumen**

Verifica el comportamiento de la base del sistema frente a grandes cantidades de datos.

En este tipo de pruebas se deberá especificar por ejemplo cual es la cantidad normal de registros con los que trabaja la aplicación y con qué volumen se realizara las pruebas, si esa información estará centralizada o distribuida, etc.

Nivel de Aceptación

Las pruebas de aceptación son el último testeo que se le realiza a la aplicación antes de que ingrese en producción. El subconjunto de pruebas que se lleven a cabo es fijado por el equipo de pruebas, de acuerdo a lo requerido por los usuarios claves, ya que son estos quienes en base a los resultados de las mismas darán su aprobación.

Usabilidad:

Capacidad para ser entendido: capacidad del producto del software que permite al usuario entender si el mismo es adecuado y cómo puede ser usado para una tarea y/o condiciones de uso particulares.

Capacidad para ser aprendido: capacidad del producto del software que permite al usuario aprender sobre su aplicación.

Capacidad para ser operado: capacidad del producto del software que permite al usuario operarlo y controlarlo.

Capacidad de atracción: capacidad del producto del software para ser atractivo al usuario.

Cumplimiento de la usabilidad: capacidad del producto del software para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad.

En el anexo C, se puede visualizar el formato del instrumento de validación de usabilidad

Confiabilidad: la confiabilidad es un atributo que mide el grado en que un producto opera sin fallas bajo condiciones establecidas por un periodo de tiempo determinado. La misma es un atributo cuantitativo que ha sido ampliamente analizado, estudiado y usado en otras industrias para caracterizar la calidad de los productos o servicios. La confiabilidad es una medida de performance a lo largo del tiempo, las misma esta expresada en función de la tasa de fallas (Hazard Failure Rate).

Análisis de los Resultados

El jefe del proyecto debe evaluar los resultados de las pruebas, analizando las incidencias recibidas y comprobando que se han llevado a cabo todos los casos de las mismas establecidos en el plan de pruebas. La evaluación consiste en los siguientes:

- ✓ Comparar los resultados obtenidos con los esperados.
- ✓ Identificar el origen de cada problema para poder remitirlo a quien proceda y determinar.
- ✓ Determinar las acciones o medidas de corrección que son precisas llevar a cabo para resolverlo de forma satisfactoria.
- ✓ Indicar que pruebas se deben volver a realizar o si es necesario contemplar nuevos casos de pruebas.

Una vez realizadas las correcciones y comprobado el correcto funcionamiento del producto de software, el jefe del proyecto documenta en el informe el resultado global de la evaluación de las pruebas

5.3 Objetivo Nro. 3: Formular una propuesta basada en las mejores prácticas de Gerencia de proyectos de software.

La importancia de hacer un plan de pruebas cada año es muy alta, pues de allí se priorizan las actividades de mayor importancia, que la empresa realizará durante el transcurso del mismo.

Utilizar un modelo o una metodología para desarrollar software, es una de las actividades más importantes en la ingeniería de software, pues a partir de esta es que se hace un desarrollo estructurado de los proyectos relacionados con software.

Es importante que la industria siga escalando en los niveles propuestos y estableciendo métricas propias que se manejen como criterios de calidad en la

empresa, para así mejorar la predicción de la realización del proceso de desarrollo de software usando técnicas, optimizando los procesos a través de mejoras continuas, incrementales y tecnológicas.

Hay determinados proyectos de desarrollo que por su propia naturaleza no pueden ser completamente definidos al comienzo del mismo, ya que requieren de un proceso de revisión y modificación constante. En este tipo de proyectos se debe utilizar una metodología de desarrollo ágil, que permita mantener la flexibilidad suficiente para adaptarse a un contexto cambiante pero manteniendo unas reglas de juego que todos los participantes en el proyecto deben conocer.

Teniendo en cuenta que mucho de los productos desarrollados sufrirán cambios durante su vida útil, y tendrán que ser adaptados a nuevos requisitos, cada vez es más necesario que existan métodos rápidos para diagnosticar defectos en el software y cumplir con la calidad deseada de dichos productos.

Aspectos relevantes a considerar

La priorización de las validaciones consiste en establecer un indicador para cada prueba que indique su relevancia. Este indicador de prioridad determinara que certificaciones deben ser ejecutadas en primer lugar y cuales son candidatas a pasar a engrosar el catálogo de pruebas de regresión o serán tomadas como posible base de las pruebas de aceptación. El aspecto de la prioridad cobra un papel importante en un proceso de generación de validaciones sistemáticas y automáticas, debido a la exposición combinatoria del número de casos de prueba.

Computacionalmente sería posible explorar un rango muy alto de combinaciones y alternativas en la ejecución de requisitos funcionales para la construcción de validaciones del sistema.

Otro aspecto relevante a considerar es el estudio del uso de estándares. Existen en la actualidad una serie de estándares ampliamente probados y utilizados que aportan un considerable valor para evitar el tener que reinventar conceptos básicos y que proporcionan una lengua común a todos los especialistas en este campo.

Metodologías Agiles

Las metodologías ágiles son flexibles, pueden ser modificadas para que se ajusten a la realidad de cada equipo y proyecto.

Los proyectos ágiles se subdividen en planes más pequeños mediante una lista ordenada de características. Cada proyecto es tratado de manera independiente y desarrolla un subconjunto de características durante un periodo de tiempo corto, de entre dos y seis semanas. La comunicación con el cliente es constante al punto de requerir un representante de él durante el desarrollo. Los proyectos son altamente colaborativos y se adaptan mejor a los cambios; de hecho, el cambio en los requerimientos es una característica esperada y deseada, al igual que las entregas constantes al cliente y la retroalimentación por parte de él. Tanto el producto como el proceso son mejorados frecuentemente.

Una metodología de desarrollo de software está compuesta por un ciclo de vida. En Ingeniería del Software existen diferentes ciclos de vida, como por ejemplo modelo en V, modelo en espiral. De acuerdo a la cronología de aparición de las metodologías ágiles, la mayoría de los métodos derivan directamente del modelo en espiral. Esto se explica ya que las dos principales características de modelo en espiral son un ciclo de vida iterativo e incremental. Por tanto, los cambios de requisitos se pueden ir integrando en cada iteración, de manera que el plan de trabajo no tiene que ser modificado, irá cambiando a lo largo de las iteraciones. Otro punto interesante es la duración de las iteraciones, con iteraciones cortas aumenta el número de reuniones con el cliente para definir y detallar sus necesidades de forma incremental.

En cuanto a la interacción de las personas, las metodologías ágiles tienden a romper las relaciones contractuales entre los clientes y los equipos de desarrollo. Esta relación se expresa en este punto de vista por el atributo de colaboración. Un equipo ágil es un tipo de organización holográfica en la que cada miembro tiene el conocimiento del sistema en su conjunto, así que, si un miembro deja el equipo, no se ha perdido conocimiento.

El principal concepto de la agilidad son los procesos ligeros. Generalmente, las metodologías ágiles incluyen menos documentación. Las pruebas son una práctica muy importante, así como la refactorización.

Manifiesto Ágil

Tras esta reunión se creó "*The Agile Alliance*", una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones a adoptar dichos conceptos. El punto de partida fue el Manifiesto para el desarrollo de software ágil (*Manifesto for Agile Software Development*, 2014). Este manifiesto se enfoca en lo siguiente:

• Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Las personas son el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

• A desarrollar software que funcione de manera adecuada y orientado a los requisitos, más que a conseguir una buena documentación. La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.

• A la colaboración con el cliente más que a la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.

• A responder a los cambios más que a seguir estrictamente un plan. La habilidad de responder a los mismos que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta. Estos valores inspiran los doce principios del manifiesto. Son

características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto a metas a seguir y organización del mismo.

La Alianza Ágil elaboró un conjunto de doce principios comunes a las metodologías ágiles de desarrollo que se enuncian a continuación:

1.- Satisfacer al cliente mediante entregas tempranas y continuas de software que le aporte un valor.

2.- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.

3.- Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.

4.- Los responsables de negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.

5.- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.

6.- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

7.- El software que funciona es la medida principal de progreso.

8.- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

9.- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.

10.- La simplicidad es esencial.

11.- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

12.- A intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Metodología *Scrum*

Su nombre no corresponde a una sigla, sino a un concepto deportivo, propio del rugby, relacionado con la formación requerida para la recuperación rápida del

juego ante una infracción menor. Su primera referencia en el contexto de desarrollo data de 1986, cuando Takeuchi y Nonaka utilizan el *rugby approach* para definir un nuevo enfoque en el desarrollo de productos, dirigido a incrementar su flexibilidad y rapidez, a partir de la integración de un equipo interdisciplinario y múltiples fases que se traslapan entre sí.

La metodología *scrum* para el desarrollo ágil de software es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos en proyectos, que emplea un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su correcto funcionamiento.

Scrum utiliza un enfoque incremental que tiene como fundamento la teoría de control empírico de procesos. Esta teoría se fundamenta en transparencia, inspección y adaptación; la transparencia, que garantiza la visibilidad en el proceso de las cosas que pueden afectar el resultado; la inspección, que ayuda a detectar variaciones indeseables en el proceso; y la adaptación, que realiza los ajustes pertinentes para minimizar el impacto de las mismas.

Los llamados equipos *scrum* son auto-gestionados, multifuncionales y trabajan en iteraciones. La autogestión les permite elegir la mejor forma de hacer el trabajo, en vez de tener que seguir lineamientos de personas que no pertenecen al equipo y carecen de contexto. Los integrantes del equipo tienen todos los conocimientos necesarios para llevar a cabo el trabajo. La entrega del producto se hace en iteraciones; cada iteración crea nuevas funcionalidades o modifica las que el dueño del producto requiera.

Scrum define tres roles: el *scrum* master, el dueño del producto y el equipo de desarrollo, tiene como función asegurar que el equipo está adoptando la metodología, sus prácticas, valores y normas; es el líder del equipo pero no gestiona el desarrollo. El dueño del producto es una sola persona y representa a los interesados, es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo; tiene entre sus funciones gestionar la lista ordenada de funcionalidades requeridas o *product backlog*. El mismo, por su parte, tiene como responsabilidad convertir lo que el cliente quiere, el *product backlog*, en iteraciones funcionales del producto; el equipo de desarrollo no tiene jerarquías,

todos sus miembros tienen el mismo nivel y cargo: desarrollador. El tamaño óptimo del equipo está entre tres y nueve personas.

Scrum define un evento principal o s*print* que corresponde a una ventana de tiempo donde se crea una versión utilizable del producto (incremento). Cada *sprint*, como en el *rugby*, es considerado como un proyecto independiente. Su duración máxima es de un mes. Un *sprint* se compone de los siguientes elementos: reunión de planeación del *sprint*, *daily scrum*, trabajo de desarrollo, revisión del sprint y retrospectiva.

En la reunión de planeación del *sprint* se define su plan de trabajo: qué se va a entregar y cómo se logrará. Es decir, el diseño del sistema y la estimación de cantidad de trabajo. Esta actividad dura ocho horas para un *sprint* de un mes. Si el *sprint* tiene una duración menor, se asigna el tiempo de manera proporcional.

El *daily scrum* es un evento del equipo de desarrollo de quince minutos, que se realiza cada día con el fin de explicar lo que se ha alcanzado desde la última reunión; lo que se hará antes de la siguiente; y los obstáculos que se han presentado. Este evento se desarrolla mediante una reunión que normalmente es sostenida de pie con los participantes reunidos formando un círculo, esto, para evitar que la discusión se extienda.

La revisión del *sprint* ocurre al final del mismo y su duración es de cuatro horas para un proyecto de un mes (o una proporción de ese tiempo si la duración es menor). En esta etapa: el dueño del proyecto revisa lo que se hizo, identifica lo que no se hizo y discute acerca del *product backlog;* el equipo de desarrollo cuenta los problemas que encontró y la manera en que fueron resueltos, y muestra el producto y su funcionamiento. Esta reunión es de gran importancia para los *sprints.*

Beneficios al utilizar la metodología scrum

- ✓ **Cumplimento de expectativas:** el cliente establece sus expectativas indicando el valor que le aporta cada requisito / historia del proyecto, el mismo los estima y con esta información el *product owner* establece su prioridad. De

manera regular, en las demos de s*print* el *product owner* comprueba que efectivamente los requisitos se han cumplido y transmite el *feedback* al equipo.

- ✓ **Flexibilidad a cambios:** alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- ✓ **Reducción del *Time to Market*:** el cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.
- ✓ **Mayor calidad del software:** la metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.
- ✓ **Mayor productividad:** se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.
- ✓ **Maximiza el retorno de la inversión:** producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.
- ✓ **Predicciones de tiempos:** mediante esta metodología se conoce la velocidad media del equipo por *sprint* (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el *backlog*.
- ✓ **Reducción de riesgos:** el hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

Metodología Proceso Racional Unificado RUP

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado visualizo la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de

desarrollo. Desde ese entonces al mando de Grady Booch, Ivar Jacobson y James Rumbaugh, Rational Software desarrolló e incorporó diversos elementos para expandir RUP, destacándose especialmente el flujo de trabajo conocido como Modelado del Negocio. En junio del 1998 se lanza Rational Unified Process. El Proceso Unificado de Desarrollo, es una metodología para el desarrollo de software orientado a objetos. El mismo es un proceso, definido como un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el proceso unificado es más que un proceso de trabajo, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones y niveles de aptitud. Está constituido por 5 flujos de trabajo fundamentales: requisitos, análisis, diseño, implementación y prueba, los cuales tienen lugar sobre 4 etapas o fases: inicio, elaboración, construcción y transición. Esta metodología es adaptable para proyectos a largo plazo y establece refinamientos sucesivos de una arquitectura ejecutable.

Características específicas de RUP

Dirigido por casos de uso: esto significa que el proceso de desarrollo sigue una trayectoria que avanza a través de los flujos de trabajo generados por los casos de uso. Los mismos se especifican y diseñan al principio de cada iteración, y son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de validaciones. Estos describen la funcionalidad total del sistema.

Centrado en la arquitectura: los casos de uso guían a la arquitectura del sistema y ésta influye en la selección de los casos de uso. La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por las plataformas de software, sistemas operativos, sistemas de gestión de bases de datos, además de otros como sistemas heredados y requerimientos no funcionales.

Iterativo e incremental: RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y las cuales se definen según el nivel de madurez que alcanzan los productos que se van obteniendo con cada actividad ejecutada. La terminación de cada fase ocurre en el hito correspondiente a cada una, donde se evalúa que se hayan cumplido los objetivos de la fase en cuestión. RUP está basado en componentes y utiliza UML para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software.

Lenguaje Unificado de Modelado (UML, *Unified Modeling Language*)

El lenguaje unificado de modelado ha ejercido un gran impacto en la comunidad software, tanto a nivel de desarrollo como de investigación. La aparición de UML ha supuesto el reconocimiento de la actividad del modelado (diseño basado en modelos) como una actividad clave para producir software de calidad.

Aunque UML no es una metodología, sino un conjunto de lenguajes, su objetivo es visualizar, específicar, construir y documentar los elementos de sistemas con gran cantidad de software. Los lenguajes definidos en UML son fundamentalmente gráficos, para facilitar su estudio y comprensión. En UML se definen nueve tipos de diagramas, algunos de los cuales modelan diferentes vistas estáticas del sistema y otros modelas vistas dinámicas:

Diagramas de clases: este es un diagrama estático en el que se muestran cuáles son las clases involucradas en el sistema y las relaciones entre ellas.

Diagramas de objetos: este otro diagrama estático está íntimamente relacionado con el anterior. Muestra una vista de las instancias reales de las clases que están en ejecución en el sistema en un momento determinado.

Diagramas de casos de uso: en estos diagramas se muestran conjuntos de casos de usos y actores y sus relaciones.

Diagramas de secuencia: estos diagramas muestran parte de la vista dinámica, concretamente una interacción entre un subconjunto de objetos, básicamente a través del envío de mensajes entre ellos. Estos diagramas priman la ordenación temporal de los mensajes.
Diagramas de colaboración: estos diagramas son isomorfos a los diagramas de secuencia, muestran la misma interacción, pero resaltando la organización estructural de los objetos.

Diagramas de estados: son máquinas de estados que especifican el funcionamiento de los objetos de una clase. Se centran en el comportamiento de sistemas reactivos dirigidos por eventos.

Diagramas de actividad: son un tipo especial de diagrama de estados que resaltan el flujo de actividad entre los objetos.

Diagramas de componentes: son diagramas estáticos que muestran la organización y las dependencias entre los componentes. El mismo suele corresponder a varias clases o interfaces.

Diagramas de despliegue: Es una vista estática estructural en la que se relacionan los nodos de procesamiento con los componentes que residen en ellos.

En UML se definen cuatro tipos fundamentales de elementos los cuales se detallan a continuación:

Estructurales: la mayoría son elementos estáticos e incluyen clases, interfaces, colaboraciones, casos de uso, clases activas, componentes y nodos.

De comportamiento: son las partes dinámicas del modelo e incluyen relaciones y máquinas de estados. Están asociados a uno o varios elementos estructurales.

De agrupación: son los elementos organizativos del modelo. Los únicos elementos de agrupación son los paquetes, que se pueden descomponer en elementos más simples.

De anotación: sirven para incluir explicaciones sobre los demás elementos del modelo. Las notas son los elementos explicativos. Son simplemente símbolos que introducen restricciones y comentarios.

Hay cuatro tipos de relaciones en UML, las cuales son:

Dependencia: es una relación semántica entre dos elementos en la que un cambio en un elemento puede afectar a la semántica en el otro.

Asociación: es una relación estructural que define un conjunto de conexiones entre objetos. La agregación es un tipo especial de asociación.

Generalización: en una relación de generalización/especialización los elementos que especializan pueden sustituir al especializado. El caso más usual en la relación de herencia entre clases.

Realización: es una relación semántica entre clasificadores en la que un clasificador especifica un contrato que otro clasificador realizaría. Esta relación se da entre interfaces y clases y entre casos de uso y diagramas de colaboración.

Representación de Lenguaje Unificado de Modelado

Lenguaje unificado de modelado, se encuentra diseñado para dibujar los esquemas en superficies bidimensionales, considerando que algunas formas corresponden a proyecciones tridimensionales en el plano bidimensional.

Hay cuatro clases de construcciones gráficas que se usan en la notación de UML: íconos, símbolos bidimensionales, rutas y cadenas.

Un icono es una figura gráfica con un tamaño y forma fijos. Pueden aparecer dentro de símbolos de área, como terminales en las rutas o como símbolos independientes que puedan o no conectar con las rutas.

Los símbolos de dos dimensiones tienen alto y ancho variables, y pueden ampliarse para permitir otras cosas tales como listas de cadenas o de otros símbolos. Muchos de ellos están divididos en compartimientos similares o de tipos diferentes.

Las rutas se conectan con los símbolos, el arrastrar o suprimir uno de ellos afecta a su contenido y las rutas conectadas. Una ruta es una secuencia de segmentos de recta o de curva que se unen en sus puntos finales.

Las cadenas presentan varias clases de información en una forma "no analizada", UML asume que cada uso de una cadena en la notación tiene una sintaxis por la cual pueda ser analizada la información del modelo. Las cadenas pueden existir como el contenido de un cuadro, como elementos en las listas, como etiquetas unidas a los símbolos o a las rutas, o como elementos independientes en un diagrama.

Sobre la base de las definiciones anteriores, UML contempla una serie de diagramas que ayudan al desarrollo. La tabla siguiente, señala los diagramas de acuerdo al área particular.

Comparación entre metodologías tradicionales vs metodologías agiles

La tabla N° 9 muestra aspectos relevantes de las metodologías de desarrollo tradicional contrastándolas con los aspectos relevantes de las metodologías de desarrollo ágil.

Tabla N° 9. Metodologías tradiciones vs metodologías Ágiles.

Metodologías tradicionales	Metodologías ágiles
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de software al finalizar el desarrollo	Entregas constantes de software
Documentación extensa	Poca documentación

De acuerdo a la tabla N° 9, tener metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta una idea interesante. Estas metodologías pueden involucrar practicas tanto de metodologías agiles como de metodologías tradicionales. De esta manera podríamos tener una metodología para cada proyecto, la problemática seria definir cada una de las prácticas, y en el momento preciso definir parámetros para saber cuál usar.

Las metodologías tradicionales se focalizan en documentación, planificación y procesos, mientras que las metodologías agiles su objetivo es la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales, respetando cronograma y presupuesto.

Es importante tener en cuenta que el uso de un método ágil no es para todos. Sin embargo, una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero y por eso las personas que no estén acostumbradas a seguir procesos encuentran estas metodologías bastante agradables

Comparación entre metodologías Scrum vs metodologías Rup

La comparación entre ambas metodologías se visualiza en la tabla N° 10 como se muestra a continuación.

Tabla N° 10. Metodología Scrum vs Metodología Rup.

Metodología Srum	**Metodología Rup**
Dirigido por evento o Sprint	Dirigido por casos de uso
Centrado Iteraciones	Centrado en la arquitectura
Iterativo e incremental	Iterativo e incremental
Orientada a personas	Orientada a Objetos

De acuerdo a la tabla N° 10**,** las metodologías agiles de desarrollo están especialmente indicadas en proyectos con requisitos pocos definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos.

Estas metodologías son imprescindibles en un mundo el cual se expone a cambios recurrentemente. Siempre el desarrollador debe tomar en consideración que lo que es la última tendencia hoy puede que no exista mañana, y por esto existe la metodología ágil donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto organizado y multidisciplinario.

5.4. Objetivo Nro. 4: Elaborar las fases del plan de gestión de pruebas para la Empresa objeto de estudio

A continuación se presentan las fases del plan de gestión de pruebas para los proyectos de software basados en las buenas prácticas de la Gerencia de Proyectos.

Plan de Gestión de Pruebas para los Proyectos de Software basado en las buenas prácticas de Gerencia de Proyectos.

Plan de Gestión Pruebas

Versión: 0.1, Fecha: 12-03-2016

Control de Versiones

Fecha de Actualización	Versión	Revisado por	Cambio / Comentario

Objetivo

Tiene como finalidad entregar los pasos a seguir para la aplicación correcta de las estrategias y pruebas necesarias en el desarrollo del software. Con el fin de verificar las funciones y procesos de los distintos módulos del software, así como también encontrar los posibles fallos o errores (bugs) que se presentan durante el periodo de pruebas. Además validar si el mismo cumple con los requerimientos que contemplen el funcionamiento total del sistema.

Al desarrollar el plan de pruebas, se puede obtener información sobre los errores, defectos o fallas que tiene el prototipo, así se realizan las correcciones pertinentes, según el caso y se asegura la calidad del producto que se está entregando al cliente. El plan de pruebas se aplica sobre el producto, es decir, el código fuente.

Alcance

Realizar las validaciones en el desarrollo del software que cumpla con los requerimientos funciones solicitadas por el cliente.

Entorno y Configuración de las Pruebas

Para el proceso de pruebas del proyecto se requiere de la disponibilidad de los siguientes entornos a saber:

a. **Requerimientos de entornos – Hardware:** lista de los requerimientos de equipos, hardware y red necesarios para completar las actividades del plan de pruebas de software. Incluye servidores de aplicación, bases de datos, equipos de PC que necesitan los *testers*, conectividad a la red (incluyendo accesos), entre otros.

b. **Requerimientos de entornos – Software:** lista de los requerimientos de software necesarios para completar las actividades de prueba, puede incluir accesos a sistemas en ambiente de pruebas y bases de datos, así como instalación de software en los computadores asignados a los *testers*.

c. **Base de Datos:** SQL Server 2008 – 2012

Criterios de Inicio

Aceptación del plan de pruebas. Revisión y aceptación del documento que contiene los casos de prueba para la certificación del proyecto.

Aceptación de paquetes. Revisión y aceptación de los software de desarrollo y que éstos cumplan con las condiciones de aceptación.

Aceptación de ambiente. Revisión y aceptación del ambiente de certificación y que éstos cumplan con las condiciones de aceptación.

Bases de Datos de Pruebas

Bases de Datos: Desarrollo

Servidor BD: RIGEL/SQL

Datos: Aleatorios

Responsabilidades y Equipo de Trabajo

Personas (Roles y Responsabilidades)

Nombre del Rol: Líder de Tecnología

Responsabilidades: es la persona asignada por la Gerencia de TI para apoyar al gerente de proyecto en la definición y ejecución de los objetivos tecnológicos definidos para el proyecto. El líder de TI, es el enlace o relación entre la estrategia tecnológica, el producto, la organización y el gerente del proyecto para garantizar los medios técnicos que permitan la eficiencia del producto o servicios.

Sus responsabilidades son:

- Asegurar la aplicación de las políticas de la empresa para el desarrollo del Software: directrices, normas, estándares y procedimientos, revisiones del Pla de Calidad del Producto, incluyendo las herramientas que se desarrollen para tal fin en el ámbito de TI.
- Gestionar plan de ambientes de desarrollo y calidad requeridos.
- Elaborar y comunicar las necesidades de adiestramiento a la PMO.
- Asegurar la alineación estratégica del proyecto con los planes de desarrollo y evolución tecnológica de la institución.
- Asegurar la ejecución de las revisiones formales, auditorías, revisiones de los planes de calidad del producto, que hayan sido planificadas y/o que considere necesarias en el ámbito de TI.
- Supervisar la ejecución física de los proyectos en lo concerniente a TI, a fin de anticipar la resolución de problemas técnicos, de equipo, de alcance o de coordinación de actividades que tengan impacto en el desarrollo del proyecto.
- Dirigir el desarrollo e implantación de proyectos de aplicaciones y/o soluciones de TI de acuerdo a los lineamientos de la PMO, con el fin de brindar soluciones que satisfagan las necesidades del cliente y que contribuyan al logro de la empresa.
- Notificar riesgos asociados a su área.

- Planificar junto al personal de desarrollo, pruebas es instalación del software en el ambiente de calidad y producción.
- Supervisar la ejecución de los Planes TI del proyecto (Alcance, costos, tiempo, calidad y riesgos) para cumplir con los requerimientos y/o productos descritos en el alcance.
- Identificar las tareas que deban incorporarse al cronograma de actividades de TI.
- Apoyar la elaboración del plan general del proyecto, definición de entregables, hitos y el alcance.
- Ejecuta los despliegues de los entregables en el ambiente de calidad.
- Participar en la elaboración de las actas tanto administrativas como del proyecto.
- Participar en la definición y análisis de los riesgos del proyecto, a fin de apoyar la toma de las acciones para mitigarlos.
- Participar en la elaboración y/o revisión de propuestas asociadas a los componentes tecnológicos del proyecto.
- Participar en la identificación de los requerimientos del proyecto, aportando su visión tecnológica de la solución.
- Liderar la evaluación de las soluciones tecnológicas.
- Participación activa en las pruebas técnicas.
- Supervisar las tareas administrativas del Plan TI.
- Gestionar la disponibilidad de la totalidad de los recursos tecnológicos que requiera el proyecto.
- Estar alineado con el Plan de Comunicación definido en el proyecto.
- Participación en los distintos comités internos del proyecto.

Etapa en las que participa: planificación, diseño, construcción e implementación.

Reporta a: Gerente de Proyecto.

Supervisa a: Equipo de TI.

Requisitos del Rol:

- Conocer los procedimientos y la plataforma tecnología de la organización.
- Ser un líder de proyectos.
- Poseer *background* técnico y funcional.
- Estar al tanto de las últimas tecnologías (innovador).
- Preparado para enfrentar diferentes problemas en su gestión.
- Habilidades de comunicación y mantener buenas relaciones interpersonales.
- Capaz de generar nuevos negocios y nuevas formas de realizar el trabajo.
- Poseer habilidades de escritura y documentación.

Nombre del Rol: Líder de Pruebas

Responsabilidades:

- Validación de Diseño Funcional.
- Generar Actualización del plan de prueba.
- Generar la matriz de casos de pruebas.
- Ejecutar los casos de pruebas.
- Registrar en la Bitácora de Evidencias los resultados de cada prueba.
- Notificación de Incidencias y registro de las mismas en la Bitácora de Registro de Incidencias.
- Documentar pruebas.
- Seguimiento y control a la generación y atención de las incidencias.
- Reporte de estatus de las pruebas.
- Coordinar con los usuarios el proceso de certificación.
- Generar informe de Cierre.

Etapa en la que participa: planificación, diseño, construcción e implantación

Reporta a: Gerente de Proyecto.

Supervisa a: No Aplica.

Requisitos del Rol:

- Experiencia en el área de *testing* y QA.
- Experiencia en el uso de alguna herramienta de pruebas y Certificaciones.

- Capacidad de Análisis, Planificación y Organización.
- Proactivo y dinámico.
- Trabajo en Equipo.
- Responsable.

Nombre del Rol: Líder de Infraestructura

Responsabilidades: la responsabilidad del líder de infraestructura dentro del proyecto, recae sobre mantener operativo la plataforma actual y facilitar la ampliación e integración de nuevos elementos que se incorporen como parte del alcance del nuevo proyecto. A continuación otras responsabilidades:

- Aplicar metodologías que sustenten la óptima operatividad y mantenimiento de la infraestructura tecnológica de la organización, así como la seguridad, respaldo y niveles de contingencia asociados a dichos componentes y su documentación.
- Planificar, coordinar y supervisar las actividades relacionadas con la administración, mantenimiento y soporte técnico de los servidores y hardware de redes que conforman la plataforma tecnológica actual de cara al proyecto.
- Supervisar las actividades que se desarrollan en los centros de procesamiento de datos y sistemas de producción.
- Controlar los eventos que se presenten y garantizar la solución eficiente para la continuidad de los servicios.
- Supervisar la administración de los ambientes de desarrollo, calidad y producción de las distintas plataformas de la organización.
- Gestionar los mecanismos y procedimientos de recuperación de las bases y estructuras de datos.
- Establecer mecanismos de almacenamiento y conservación de los datos antes, durante y después de la ejecución del proyecto.
- Supervisar la ejecución de los diferentes controles de cambios en la plataforma tecnológica.
- Asegurar la continuidad de los servicios, así como la atención oportuna a los usuarios.

Etapa en la que participa: diseño, construcción e implementación.

Reporta a: Líder de Tecnología.

Supervisa a: No Aplica.

Requisitos del Rol:

- Conocimiento del mantenimiento de áreas de plataforma de sistemas.
- Poseer alguna certificación en redes y telecomunicaciones (CCNA, CCNP, CompTIA A, ISO/IEC 27001).
- Certificación en plataforma de telecomunicaciones, MCSE (deseable).
- Habilidades de negociación.
- Supervisión de personal.
- Manejo de grupos e indicadores de gestión.

Nombre del Rol: Líder de Sistemas de Información (Arquitecto de Software)

Responsabilidades: el arquitecto de software es el responsable de analizar/evaluar las necesidades y/o requerimientos de la organización con respecto al componente de software de la solución tecnológica planteada en el alcance del proyecto. Además, debe realizar las estimaciones de esfuerzo y tiempo necesarios para los desarrollos.

El arquitecto debe hacer uso de sus habilidades técnicas (“duras”) y no-técnicas (“suaves”) e identificar estilos arquitectónicos y tecnologías que sean apropiados para resolver los problemas y/o necesidades del negocio y proponer una solución.

Entre otras, las responsabilidades son:

- Implementar metodologías de desarrollo de software.
- Proveer guías, documentaciones y técnicas de desarrollo de software al equipo de proyecto.
- Identificar el alcance y restricciones de un proyecto.
- Alinearse con la visión de la organización en función a la implementación de los proyectos.
- Prestar asesorar en la definición y planificación de los proyectos de desarrollo de software.

- Definición de estándares y políticas de desarrollo y reutilización de componentes y servicios.
- Evaluación y análisis de las necesidades del cliente en función al alcance del proyecto.
- Definición de la Arquitectura de Software a desarrollarse o implementarse.
- Facilitar los procedimientos para la selección de tecnologías y software.
- Realizar las evaluaciones continuas sobre las soluciones de software existentes y plantear las mejoras continuas de la arquitectura.
- Responsable del aseguramiento de la calidad de software a desarrollar.
- Preparación del plan de pruebas de proyectos de desarrollo de sistemas de información.

Etapa en la que participa: planificación, diseño, construcción e implementación.
Reporta a: Líder de Tecnología.
Supervisa a: No Aplica.
Requisitos del Rol:

- Amplios conocimientos técnicos y funcionales de la arquitectura de software existente de la organización.
- Gran experiencia en programación y desarrollo de software.
- Conocimiento de Mapeo de Objetos a RDBMS.
- Conocimiento Integración de API's, *Web Services* y otros componentes.
- Conocimiento de la Reutilización y *Frameworks*.
- Conocimiento en Lenguajes de Programación Orientado a objetos.
- Conocimiento en manejadores de bases de Datos SQL Server.
- Conocimiento amplio en herramientas .net.
- Amplio conocimiento sobre utilización de metodologías de desarrollo de software.
- Conocimiento de tecnologías y protocolos como: SOA, SOAP, XML, WSDL, HTTP, HTTPS, TCP/IP, etc.
- Liderazgo y Formador.

Nombre del Rol: Líder de Interfaces

Responsabilidades: el líder de Interfaces es quien se responsabiliza de garantizar el flujo de información y continuidad de las operaciones y transacciones que se realizan entre máquinas o elementos de computación, informática y telecomunicaciones.

Las responsabilidades son:

- Analizar los requerimientos del proyecto y determinar las interfaces.
- Analizar y comprender los elementos, componentes, objetos y servicios. Luego, definir las tareas propias de la implementación del intercambio de información (interfaz).
- Determinar las interfaces de los Sistemas, Servicios Web, y otros.
- Definir Interface de software/hardware, software/base de datos, software/usuario, base de datos/base de datos, hardware/base de datos, otros.
- Validar Diseños de las interfaces con la arquitectura de software y hardware.
- Diseño de estructura (objetos y acciones) de la interfaz.
- Documentación de los procesos de interface.
- Preparar Documento de Diseño de interfaz (completo).
- Manejo de herramientas de soporte para interfaces y protocolos.
- Definir estrategias de recuperabilidad, donde la interfaz debe incluir mecanismos recuperarse de las fallas y errores. Además, debe proveer retroalimentación significativa.
- Asegurarse que la interfaz debe proveer características de interacción apropiada para los diferentes tipos de componentes y usuarios del sistema.

Etapa en la que participa: planificación, diseño, construcción e implementación.

Reporta a: Líder de Tecnología.

Supervisa a: No Aplica.

Requisitos del Rol:

- Conocimientos técnicos y funcionales de la arquitectura de hardware y software existente.
- Experiencia comprobable en programación y desarrollo de software.

- Conocimiento de Manejadores de Base de Datos SQL Server y otros.
- Conocimiento de ingeniería de Software, API's, Dll, *Web Services* y otros componentes.
- Conocimiento de tecnologías y protocolos como: SOA, SOAP, XML, WSDL, HTTP, HTTPS, TCP/IP, etc.
- Liderazgo y Formador.

Criterios de Aprobación o Rechazo

- Errores Graves: información crítica presentada erróneamente, información mal registrada en la base de datos, caídas de programas, incumplimiento de objetivos en funciones principales, entre otros.
- Errores Medios (comunes): bug en presentación de datos, incumplimiento de objetivos en funciones secundarias, caídas de programas auxiliares, entre otros.
- Errores Leves: errores en presentación de datos secundarios, no adecuación a estándares, comportamientos correctos pero diferentes en situaciones similares, dificultades de operación, entre otros.

Los criterios de aprobación se muestran en la tabla N° 11, como se visualiza a continuación:

Tabla N° 11. Criterios de Aprobación.

Criterio	Descripción
Aprobado	Se aprobara el software con un 100% de las pruebas ejecutadas, pero con un 90% de aceptación. Esto quiere decir que el 90% de las pruebas deben ser exitosas y sin errores. El restante 10% pueden existir errores medios o bajos pero no graves.
Rechazado	En caso de ocurrir que el software no cumpla con el nivel exigido, el proyecto se rechaza completo en su etapa de certificación.

Registro de los Resultados de las pruebas

Pruebas de Componentes

En la tabla N° 12 se visualizan las pruebas de componentes.

Tabla N° 12. Pruebas de Componentes del Software.

<table>
<tr><td>Objetivos de la Prueba</td><td>Comprobar el módulo de Inventario de registros de mercancía.</td></tr>
<tr><td>Técnicas</td><td>Ingresar al formulario del registro de inventario de mercancía datos correctos de un producto y datos falsos para comprobar su funcionamiento.</td></tr>
<tr><td>Casos de Prueba</td><td>Formato de Casos de Prueba

Tipo de Prueba: Unitaria (Caja Blanca).
Objetivo: Validar si el registro se guarda correctamente al agregar los datos correspondientes.

Caso Nro. 1

Descripción: Datos correcto. Información de un Producto

Entradas: Código, descripción, tipo de Servicio, tipo de IVA, proveedor, peso, línea, sublinea, unidad, precio, unidad etc.

Salidas Esperadas: Registro exitoso.</td></tr>
<tr><td>Herramientas Requeridas</td><td>• Debuggers.
• Manejador de Base de Datos SQL.
• Tracing y Seguimiento a variables.</td></tr>
<tr><td>Resultados</td><td>Todos los casos de pruebas planificados se han ejecutado.
Todos los defectos identificados se han considerado.</td></tr>
<tr><td>Observaciones</td><td></td></tr>
</table>

Prueba de Sistemas

Pruebas Funcionales

En la tabla N° 13, tabla N° 14, tabla N° 15, tabla N° 16 se visualizan las pruebas funcionales del software.

Tabla N° 13. Pruebas Funcionales de Interfaces del Software.

Objetivos de la Prueba	Comprobar el funcionamiento del registro de datos en los formularios del sistema, comprobar las interfaces, Web Service, que trabajen de manera eficiente.
Técnicas	Evaluar la funcionalidad de las formas, formularios, interfaces, Web Service, introduciendo datos validos e inválidos.
Resultados	Todas las pruebas planificadas se han ejecutado. Todos los defectos identificados se han considerado.

Tabla N° 14. Pruebas Funcionales de Casos de Uso del Software.

Objetivo de la Prueba	Navegar por las opciones modificar, eliminar, buscar, barras de desplazamiento para verificar que cada una de ellas lleva al proceso deseado.
Técnicas	• Cuando se seleccione modificar el dato deseado, cambie automáticamente al guardar los cambios. • Al eliminar un dato seleccionado se debe borrar del sistema permanentemente. • Debe contener búsqueda asistida.
Caso de uso involucrado	Caso de uso Modificar, eliminar, buscar datos.
Resultados	Todas las pruebas planificadas se han ejecutado. Todos los defectos identificados se han considerados.

Tabla N° 15. Pruebas Funcionales de Reglas del Negocio del Software.

Objetivo de la Prueba	• Validar los procesos y reglas de negocio establecidas. • Validar que se cumplan los requerimientos funcionales establecidos • Validar los requerimientos mínimos para el funcionamiento de la aplicación.
Técnicas	• Los resultados esperados ocurren cuando se usan datos válidos. • Se despliegan mensajes de error cuando se usan datos inválidos. • Cada regla de negocio es propiamente aplicada. • Realizar set de pruebas de los requerimientos mínimos para el adecuado funcionamiento de la aplicación
Resultados	• Acta de aprobación de la ejecución de las respectivas pruebas. • Resultados de la ejecución de los scripts de pruebas. • Análisis de los defectos encontrados durante el proceso de pruebas. • Análisis de las pruebas de regresión y solicitud de correctivos en caso que se presenten defectos durante estas pruebas

Tabla N° 16. Pruebas Funcionales de Módulos del Software.

Objetivo de la Prueba	Validar la integración entre los diferentes módulos que componen la solución con el fin de garantizar que su operación integrada es correcta.
Técnicas	• Combinación de módulos de bajo nivel en grupos que realicen una misma función o subfunción específica, con el fin de reducir el número de pasos de integración. • Se escribe para cada módulo un módulo impulsor o conductor, con el fin de simular la llamada a los módulos, introducir datos de pruebas y recoger resultados.
Resultados	• Acta de aprobación de la ejecución de las respectivas pruebas que contenga: • Resultados de la ejecución de los scripts de pruebas. • Análisis de los defectos identificados durante el proceso de pruebas, considerados

Pruebas de Aceptación

Pruebas de Usabilidad

Para realizar las pruebas de usabilidad se utilizó la tabla N° 17 con una leyenda donde se muestra la valoración y las 10 heurísticas de Nielsen.

Tabla N° 17. Pruebas de Usabilidad del Software.

Leyenda de Valoración	Heurística de Nielsen
0 – No es un problema de usabilidad. 1 – Problema menor. 2 – Problema mayor de usabilidad, importante fijar solución. 3 – Usabilidad catastrófica, imperativo fijar solución.	H1: Dialogo natural y simple. H2: Hablar el lenguaje de usuario. H3: Minimizar la carga cognitiva. H4: Consistencia. H5: Feedback. H6: Proveer claramente las salidas. H7: Proveer Shortcuts. H8: Mensaje de error descriptivos. H9: Prevención de errores. H10: Asistencia al usuario.

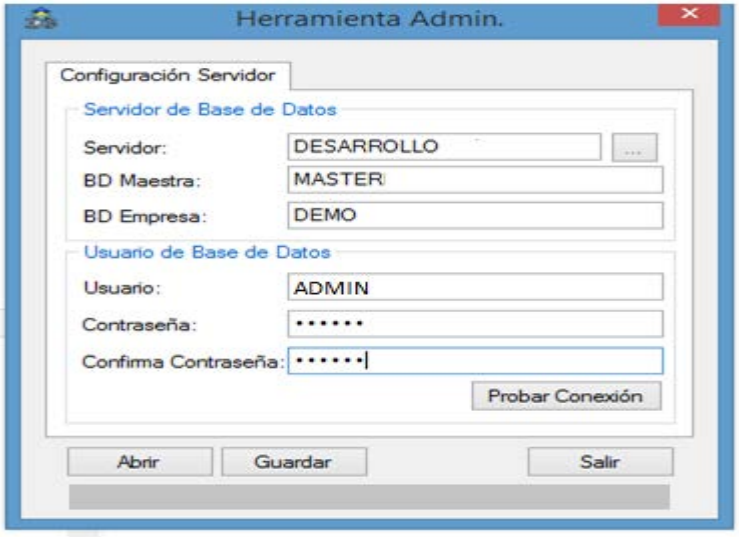
Pruebas Alfa

Las pruebas alfa del software se visualizan en la tabla N° 18, tabla N° 19 como se muestra a continuación.

Tabla N° 18. Pruebas de Alfa del Software.

Objetivos de la Prueba	Al ingresar los datos correctos al sistema, el usuario pueda acceder de manera apropiada a la aplicación
Técnicas	• Cuando se ingresan los datos correctos, automáticamente accede al sistema. • Cuando se escriben datos incorrectos muestra un mensaje de error y es posible acceder. • Dependiendo del tipo de usuario se desplegara un menú diferente al ingresar al sistema.

Tabla N° 19. Pruebas de Alfa del Software.

Objetivos de la Prueba	Al ingresar los datos correctos a la aplicación, el administrador de la aplicación puede registrar exitosamente la configuración del servidor
Técnicas	• Al ingresar los datos correctamente en el formulario se registra la configuración del servidor en la base de datos. • Al dejar un campo vacío que es obligatorio, no permite el registro ya que están validados los campos y aparecerá un mensaje para que llene determinado campo.
Interfaz Asociada	
Herramientas Requeridas	• Manejador de Base de Datos SQL. • Debuggers. • Tracing y Seguimiento a variables. • .Net Visual Studio.
Resultados	• Acta de aprobación de la ejecución de las respectivas pruebas que contenga: • Resultados de la ejecución del ingreso de los datos, como funciono la interface.
Consideraciones	Para realizar este proceso debe ser administrador del manejador de base de datos.

Pruebas de regresión

Las pruebas de regresión del software se visualizan en la tabla N° 20, como se muestra a continuación.

Tabla N° 20. Pruebas de Regresión del Software.

Objetivos de la Prueba	Validar que el sistema siga funcionando perfectamente después de que las acciones correctivas y/o realces son aplicados.
Técnicas	• Repetir las pruebas (unitarias, de integración, funcionales y de carga) que se hicieron antes de corregir defectos o de añadir nuevas funcionalidades, para comprobar que las modificaciones no provocan errores donde antes no los había.
Herramientas Requerida	• Casos de Prueba • Bug Tracker • Tracing
Resultados	• Acta de aprobación de la ejecución de las respectivas pruebas • Resultados de la ejecución de los scripts de pruebas • Análisis de los defectos encontrados durante el proceso de pruebas • Análisis de las pruebas de regresión y solicitud de correctivos en caso que se presenten defectos durante estas pruebas

Análisis de los resultados de las pruebas

El análisis de los resultados de pruebas se plasmó en la tabla N° 21, como se visualiza a continuación.

Tabla N° 21. Análisis de resultados de las pruebas.

Tipo de Prueba	Casos de Prueba	Resultado	Criterio de Finalización
Regresión	Validar nuevamente después de las correcciones	Esperado	Aprobado
Funcionales	Todos los módulos han sido probados	Esperado	Aprobado
Interfaz de Usuario (Usabilidad)	Los problemas de usabilidad fueron corregidos gracias a la aplicación de las heurísticas de Nielsen	Esperado	Aprobado
Alfa	Todas las funcionalidades del módulo se ajustan a los requerimientos solicitados	Esperado	Aprobado

Procedimientos de Usuario

Para utilizar la herramienta de manera adecuada se necesitan guías o manuales que sean claros, correctos, completos y coherentes, para que el usuario pueda manejar la herramienta de forma correcta y pueda comprender los conceptos tras la funcionalidad. A continuación se muestran los diferentes atributos de calidad de estos procedimientos:

- **Clara**: las instrucciones proporcionadas en el documento, deben ser lo suficientemente explicitas para que el usuario pueda desenvolverse dentro del entorno de la herramienta.
- **Correcta**: no existen errores semánticos, sintácticos, ortográficos ni de enlace dentro de la documentación proporcionada al usuario.
- **Completa**: la información debe estar completa, desde la parte técnica hasta la parte funcional.
- **Coherente**: no existen ambigüedades, ni incongruencias dentro del documento que puedan confundir al usuario.

Los documentos a entregar con la herramienta se detallan a continuación:

A. Manual de Usuario. manual que tiene las instrucciones para poder manejar la herramienta sin problemas.
B. Manual de Instalación. manual que tiene las instrucciones sobre cómo instalar la herramienta.

El manual de aseguramiento de la calidad es con el cual se puede controlar que los procesos y el producto de ellos, cumplen con lo sugerido en las mejores prácticas.

Riesgos asociados del procedimiento de pruebas

Los riesgos asociados al procedimiento de las pruebas del software se visualizan en la tabla N° 22, como se muestra a continuación.

Tabla N° 22. Matriz de Riesgos del procedimiento de las Pruebas.

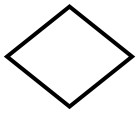
Riesgo Identificado	Probabilidad de Ocurrencia	Impacto	Medidas de prevención – Plan de respuesta al riesgo
No se cuenta con un ambiente real de infraestructura antes del primer ciclo.	Media	Alto	Configurar un ambiente con características similares, a la infraestructura real.
Identificar tardíamente problemas de compatibilidad con plataformas externas de alto riesgo o costo.	Media	Alto	Cubrir en las pruebas una cantidad representativa de plataformas que deben ser compatibles con la solución a futuro.
No incluir en las listas de chequeo de comprobación de los requerimientos, los aspectos relacionados con la facilidad de operación, por desconocimiento en los mismos.	Alta	Alto	Detectar a tiempo aspectos del diseño que se conviertan en impedimentos para permitir la fácil instalación y la administración de la solución.

Diagrama de Flujo de Procesos

En el anexo e, se logra visualizar el diagrama de flujo del proceso de pruebas. Los componentes usados para modelar este diagrama son:

Inicio del Flujo

Actividad



Decisión

Fin del Flujo

Relación o Flujo de las Actividades

CAPITULO VI: ANALISIS DE LOS RESULTADOS

El presente capítulo tiene por finalidad realizar la evaluación de los resultados obtenidos en este proyecto, en función de los objetivos planteados en el capítulo I del trabajo evaluado. Se analiza el cumplimiento de los objetivos específicos y el logro del objetivo general del mismo.

Planificar: Definiciones Acciones y Procedimientos a Ejecutar en caso de detectar no Conformidades

Las empresas deben enfrentar nuevos retos que implican mejorar la calidad de los productos y servicios brindados. Sin embargo el termino calidad puede llegar a tener diferentes interpretaciones dependiendo de la persona o la empresa que lo esté valorando, por lo cual es importante tener un estándar sobre los requisitos que se desean cumplir.

La norma exige que se redacten procedimientos para las acciones correctivas y para las acciones preventivas. Es aconsejable que ambos procedimientos separados, para así poder discernir mejor la diferencia entre acción correctiva y acción preventiva.

Planificar el Seguimiento y Control

Cuando se detecta una no conformidad, se debe llevar a cabo la corrección del fallo y analizar la causa del mismo, a continuación tiene que registrarse documentalmente, para lo cual el procedimiento de no conformidades, contiene el diseño de un documento destinado a éste fin.

Las no conformidades deben quedar registradas en un documento numerado y fechado en el día en que se cumpla

La numeración debe contemplar las diferentes tipologías de fallos y asimismo debe identificar los diferentes procesos. No obstante los tipos y procesos a los que corresponda la no conformidad pueden identificarse de cualquier otra forma en el

formulario.
El documento para registrar la no conformidad, deberá contener los siguientes datos y requisitos:

Fecha en que se detecta la no conformidad (fecha e incluso hora).

Descripción de la no conformidad: debe ser detallada, clara, concisa y corta, incluyendo lo que ha pasado, medido, etc., la fecha y lo que la norma establece como requisito. Es algo tan simple como indicar exactamente cuál es el requisito a cumplir según la norma, cual es el dato real y la fecha en que se ha producido.

Explicación de la No Conformidad: es decir una declaración escrita de la no conformidad. Se debe explicar cuál ha sido el motivo o la causa que supuestamente ha dado lugar al incumplimiento del requisito.

Medida correctiva propuesta: exponer qué medida tomar ante el incumplimiento que se ha detectado.
Fecha en que se pone en marcha la medida correctiva, plazo de tiempo que deberá pasar para comprobar si la corrección propuesta ha dado resultado, verificación y firma por el responsable, de revisar si la medida correctiva propuesta ha dado resultado.

Cierre de la No Conformidad: una vez se ha verificado y confirmado que la acción correctiva ha sido la correcta, se procede al cierre de la no conformidad, por el responsable en cuestión, mediante su firma y con indicación de la fecha de cierre.

Medida preventiva: acción que se realiza para evitar que se produzca una no conformidad potencial u otra causa no deseable. Debe controlarse igualmente, fecha en que se propone, fecha de implantación, verificación, cierre.

Gestión de Requisitos

Los requisitos del sistema son especificados en el Documento de Especificación de Requerimientos del Sistema. Cada requisito tendrá una serie de atributos tales como prioridad, estado, iteración donde se implementa, etc. Estos atributos permitirán realizar un efectivo seguimiento de cada requisito. Dichos requerimientos serán manejados en el gestor de configuración y cualquier cambio que en alguno de ellos se realice deberá primero ser consultado al resto de integrantes del grupo.

Control de Calidad

Los defectos detectados en el software entraran a una lista de elementos por corregir con nivel de prioridad cada uno de estos defectos tendrán un tiempo máximo de corrección se le informará a cada miembro del equipo el papel que debe desempeñar en la corrección y detección de errores.

Reporte de No Conformidad

De acuerdo a los estándares de ISO 9000 una no conformidad se presenta cuando un requisito del producto o del servicio no es cumplido. Una no conformidad puede ser detectada de varias formas, por ejemplo durante una auditoría realizada por el Sistema de Gestión de Calidad, por la queja de un cliente o durante una inspección de calidad dentro del proceso.

Si una no conformidad es encontrada, el auditor o la persona que realizó el reporte deben incluir la información respecto a ella, por ejemplo descripción, él o los requerimientos que están siendo incumplidos, el área que debe resolver el problema, etc. La información suministrada debe ser suficiente para que se logre detectar y corroborar la no conformidad. De acuerdo a lo anterior expuesto el reporte de no conformidad se puede visualizar en el anexo b.

Alcance

Realizar las validaciones correspondientes en el proceso de dar solución a las No Conformidades encontradas, está diseñado para corregir las no conformidades y eliminar sus causas.

También es posible efectuar una acción preventiva para evitar no conformidades potenciales.

Objetivos

- ✓ Implementar soluciones inmediatas para una no conformidad.
- ✓ Realizar Acciones Correctivas para eliminar las causas vitales de una no conformidad.
- ✓ Realizar Acciones Preventivas para eliminar las causa vitales de una no conformidad potencial.

Grado de Aceptación

El grado de aceptación termina de definir el nivel de calidad de un software y le permite conocer al equipo de desarrollo qué tan bien supo interpretar los pedidos del cliente durante la gestación del proyecto.

Conocimiento Técnico

Tiene una finalidad, está orientado al saber hacer, a crear objetos artificiales que tienen una finalidad práctica, a satisfacer sus necesidades modificando la naturaleza.

El conocimiento técnico es la medula espinal de los procesos tecnológicos, gracias a los cuales el hombre puede satisfacer sus necesidades e intereses.

Métricas del desarrollo del Software

El producto debe cumplir con las expectativas del cliente, pero sobre todo debe ser un elemento controlador de las mismas, debe proporcionar información que el cliente sienta que son claros sus alcances y están acordes con su trabajo.

Las métricas del desarrollo del software se visualizan en la tabla N° 23, como se muestra a continuación.

Tabla N° 23. Métricas del desarrollo del Software.

Actividad	**Métricas**
Gestión de Requisitos	• Cantidad de casos de Usos Obtenidos. • Cantidad de Requisitos abarcados en los casos de usos. • Cantidad Horas/Hombres consumidos en la actividad. • Cantidad de errores o no conformidades por cada entregable
Análisis	• Cantidad de casos de usos realizados. • Cantidad de prototipos diseñados. • Cantidad de clases de análisis. • Cantidad de Horas/Hombres consumidos en la actividad. • Cantidad de errores o no conformidades por cada entregable. • Cantidad de no conformidades del cliente con el modelo de análisis.
Diseño	• Cantidad de clases. • Cantidad de Horas/Hombres consumidos en la actividad. • Cantidad de errores o no conformidades por cada entregable. • Cantidad de no conformidades del cliente con el modelo de diseño.
Codificación	• Cantidad de requisitos abarcados. • Cantidad de casos de uso implementados. • Cantidad de Elementos de Código (clases, métodos, etc.) documentados • Cantidad de Componentes reutilizables generados • Cantidad de Horas/Hombre consumidas en la actividad.
Pruebas	• Cantidad de Casos de Prueba diseñados. • Cantidad de Casos de Prueba ejecutados. • Cantidad de Casos de Prueba ejecutados con errores graves. • Cantidad de Casos de Prueba ejecutados con errores leves. • Cantidad de Casos de Prueba ejecutados con errores de cosmética. • Cantidad de Horas/Hombre consumidas en la actividad.

CAPITULO VII: LECCIONES APRENDIDAS

El presente Trabajo Especial de Grado presentó un diseño de un plan de gestión de pruebas de software basado en las mejores prácticas de la gerencia de proyectos. Sobre la base del estudio realizado, se puede decir que se dio respuesta a cada uno de los objetivos específicos planteados.

Aun cuando las empresas desarrollen sus actividades de manera exitosa, existen oportunidades de mejora para sus procesos, que permitirán llevarlos a cabo de la manera más eficiente, por ello es muy importante dejar plasmado un documento de lecciones aprendidas que, contribuyen a transformar el conocimiento tácito o entendimiento (aquel que proviene de la mente y de las experiencias) en conocimiento explicito (aquel que se plasma en documentos, registros, archivos u otros) para su difusión.

A fin de contribuir a la difusión de las experiencias obtenidas durante el desarrollo del trabajo especial de grado se expondrán a continuación los aspectos más relevantes que dieron oportunidad a un mayor conocimiento y mejorar e interpretar los resultados de la investigación realizada. En la tabla N° 24, se visualizan las lecciones aprendidas del Trabajo Especial de Grado como se muestra a continuación.

Título del proyecto: Diseño de un Plan de Gestión de Pruebas de Software basado en las mejores prácticas de la Gerencia de Proyectos.

Tabla N° 24. Lecciones Aprendidas.

Lecciones Aprendidas	Descripción, Impactos, Soluciones
Tener un equipo técnico experimentado y diverso	• Desarrollo de diseños y especificaciones detalladas con pocos cambios en el diseño. Esto dio lugar a un coste considerable y el ahorro de horario • Una mejor estimación de los costos desarrolló debido a un diseño más detallado • Superar más fácilmente retos técnicos, debido a la experiencia y el conocimiento de los miembros del equipo
Mejorar el control con los usuarios funcionales	• Un proyecto de software no es únicamente responsabilidad del equipo de sistemas. • Los usuarios funcionales deben involucrarse en la pruebas y en el análisis de las potenciales mejoras para sus procesos de negocio de las nuevas funcionalidades. • Hay cambios sobre funcionalidades existentes y nuevos módulos y funcionalidades con los que se deben familiarizar.
Realizar regularmente estimación de abajo hacia arriba para completar	• Este ejercicio fue extremadamente útil en el mantenimiento de la integridad del plan de proyecto, para centrar la atención de los interesados en el proyecto a los problemas potenciales del proyecto, y para la realización de las revisiones del software. • Este proceso también proporciona una verificación independiente del trabajo realizado, y verificó la exactitud de las herramientas mensuales de seguimiento de proyectos.
Mejorar el control de calidad y pruebas de los procesos del software	• La empresa tiene la política general de control de calidad y pruebas de componentes; sin embargo, los procedimientos de control de calidad y pruebas específicas del proyecto eran inconsistentes y demasiado general. Como resultado, algunos procesos del software tenían pruebas de control de calidad, mientras que muchos no fueron sometidos a los mismos. • Debido a que algunos procesos se comprobaron efectivamente y se establecen correctamente en la matriz de pruebas antes de la corrida del ejecutable para posterior distribución a los clientes.
Asesores de los procesos de software	• Como una práctica de lista de chequeo (Checklist). • Antes y durante una evaluación de procesos de software.
Documentación de las lecciones.	• Es necesario documentar las lecciones aprendidas del diseño de un plan de gestión de pruebas de software basado en las mejores prácticas de la gerencia de proyectos para dar aporte al trabajo realizado y permitir a otros tomarlas en cuenta en sus propios proyectos.

CAPITULO VIII: CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones y recomendaciones identificadas durante el desarrollo del presente Trabajo Especial de Grado, en el que se propone el Diseño de un Plan de Gestión de Pruebas de Software basado en las mejores prácticas de la Gerencia de Proyectos.

Conclusiones

De acuerdo a lo planteado en el presente Trabajo Especial de Grado y para dar respuesta a los objetivos planteados se puede concluir en lo siguiente:

Objetivo Nro. 1: Caracterizar los proyectos de Software de la empresa en estudio.

- Como resultado de esta investigación se caracterizaron los proyectos de software de la empresa en estudio, destacando que el software desarrollado por la empresa ofrece los beneficios de la integración del manejador de base de datos SQL, y las ventajas de la tecnología cliente/servidor, pero es necesario establecer actividades que se desarrollen para garantizar la calidad del software en el proceso de validación y certificación del mismo.
- Con el software diseñado se obtiene un sistema de información apto para una empresa ya que proporciona características tan avanzadas como: procesamiento de transacciones a alta velocidad e integridad de datos, seguridad, escabilidad y flexibilidad modular.
- El software desarrollado luego de implantado en el cliente también puede ser adaptado a la medida de las necesidades de la empresa, permitiendo gestionar de forma optimizada, organizada la empresa o negocio.

Objetivo Nro. 2: Analizar las mejores prácticas en el área de gestión de pruebas para proyectos de Software.

- La idea general que se tiene para validar aplicaciones o sistemas informáticos es realizar un conjunto de pruebas de diferente índole y tipo, desarrolladas por un conjunto de personas expertos en la empresa validan los requisitos funcionales del software a crear.
- Por otro lado, la prueba beta es la que se le proporciona a usuarios finales, situados en lugares concretos de los puestos de trabajo donde finalmente será el software implantado, para que sea otra vez el usuario y sin contar con la presencia del equipo de desarrollo, el que de nuevo vuelva a emitir unos informes de resultados e impresiones de la aplicación o sistema software desarrollado.
- Independiente del proceso seguido, es fundamental que la calidad tanto del código como de la documentación aportada al usuario sea alta y se corresponda con los parámetros adecuados para el software que concretamente se esté desarrollando.

Objetivo Nro. 3: Formular una propuesta basada en las mejores prácticas de Gerencia de proyectos de software.

- La mayoría del personal de calidad, desarrollo ágil de software trae de vuelta el control de la calidad en el análisis y diseño del producto, manteniendo control de calidad muy involucrado en la toma de decisiones en todo el ciclo de vida completa.
- Teniendo en cuenta la importancia de las pruebas se puede resumir como actividades esenciales para su buen desarrollo la planeación, el desarrollo y el reporte de las pruebas. De estas actividades se debe dejar una evidencia como soporte de su realización.
- La utilización de una norma internacional como la IEEE 829 como base del diseño de la aplicación, permite que los soportes de las pruebas de los desarrollos que se estén realizando tengan concordancia con la

documentación exigida internacionalmente, pero que a su vez sea flexible y adaptable a las necesidades y restricciones de desarrollos particulares de los grupos de investigación.

Objetivo Nro. 4: Elaborar las fases del plan de gestión de pruebas para la empresa objeto de estudio.

- Se elaboraron las fases del plan de gestión de pruebas para los proyectos de software, el cual contiene el documento del plan de pruebas aplicar para la evaluación y certificación de las pruebas del software.
- Se identificaron algunos riesgos que deben ser tomados en cuenta durante la validación y certificación del software, ya que pueden retrasar el mismo.

Recomendaciones

Luego de haber analizado los datos y diseñar un Plan de Gestión de Pruebas de Software basado en las mejores prácticas de la Gerencia de Proyectos, se hace necesario plantear las siguientes recomendaciones en función de lograr mejoras en la organización, en sus procesos de desarrollo del software.

- Capacitar al personal de la GTI en el área de formación de equipos de alto desempeño, con la finalidad de aprovechar el interés observado durante la investigación frente a las posibles mejoras en los procesos de gestión de proyectos, para alinearlos hacia la aplicación de mejores prácticas mediante el trabajo en equipo y crear motivación entre los integrantes del equipo.
- Es importante validar todos los procesos que el cliente ejecuta de forma manual con la finalidad de adaptarle todos los procesos con la puesta en marcha en la implantación del software.

- Reforzar políticas de acercamiento a los clientes y evaluar los tiempos de respuestas a las incidencias reportadas por medio del sistema integral sigma, aportando soluciones.

- El reconocimiento será un factor clave para fortalecer dicha dinámica y cultura innovadora, por lo cual, se deben propiciar condiciones para que todos los colaboradores se movilicen hacía la innovación y la apropiación de conocimiento, por medio de la capacitación en los procesos que conforman el sistema y en los puntos específicos para fomentar la creatividad e investigación y así se fortalezca la capacidad creativa de la organización, es decir, inteligencia colectiva con el único propósito, el de cubrir las necesidades y deseos, a través del desarrollo de nuevos o significativamente mejorados productos y servicios.
- Incentivar el uso de las mejores prácticas de Gerencia de Proyectos en el Departamento de Gerencia de Tecnología de la Información GTI.
- Se recomienda la actualización de la metodología en el transcurso del tiempo ya que lo procesos internos y las prácticas en IT van cambiando incluso evolucionando.
- Fortalecer y consolidar al equipo de desarrollo del software a través de la adquisición de conocimientos relativos a las mejores prácticas, procurando así establecer un proceso de formación continua dentro y fuera de la organización que permita instruirse y poder incrementar los conocimientos y competencias en el desarrollo y validación del software, creando motivación entre los integrantes del equipo.
- Se identificaron algunos riesgos que deben ser considerados durante la ejecución del plan, ya que pueden retrasar el mismo.
- Establecer indicadores de gestión del control de la calidad del software e implementar mecanismos para su seguimiento y control.

Finalmente, como se puede apreciar en lo anteriormente expuesto, se cumplió con el objetivo general planteado en el presente Trabajo Especial de Grado, ya que se propuso Diseñar un Plan de Gestión de Pruebas para los Proyectos de Software basado en las buenas prácticas de Gerencia de Proyectos.

REFERENCIAS BIBLIOGRAFICAS

Arias, F. (2012). El Proyecto de Investigación: Introducción a la metodología científica. Sexta Edición. Caracas: Episteme.

Balestrini, M. (1998) Como se Elabora el Proyecto de Investigación. Segunda Edición. Caracas. B.L Consultores Asociados.

Bedini, A.; Guerra, L. (2005) Gestión de Proyectos de Software. Universidad Técnica Federico Santa María.

Beth, M.; Konrad, M.; Shrums, S., CMMI (2009). Guía para la integración de procesos y la mejora de productos, 2da. ed., Madrid, Editorial Pearson Educación.

Chamoun, Y. (2002). Administración Profesional de Proyectos La Guía. México: McGraw-Hill

CII (1.996). El manual de Constructibilidad, Preparedfor Construcción Instituto Industria, Australia. Informe de Investigación 8, abril

Colegio de Ingenieros de Venezuela (1996) Código de Ética Profesional. Recuperado el 20 de Octubre de 2015, de http://www.civ.net.ve/uploaded_pdf/cep.pdf.

Constitución de la República Bolivariana de Venezuela (Gaceta Oficial Extraordinaria N° 5.453 de fecha 24/03/2000).

Environmental Management Project Definition Rating Index (EM-PDRI), Revision 1, EM- DOE, 2001. USA.

Escobar, J. y Cuervo, A. (2008) Validez de contenido y juicio de expertos: una aproximación a su utilización. Avances en Medición, 6, 27–36. Colombia.

Estudio de Mercado de Servicios i + e Software en Venezuela (2013). Disponible en: http://www.cavedatos.org.ve

Evans, J. William, L. (2008). Administración y control de calidad 7ª Edición. México: International Thomson.

Eyssautier, M. (2008). Metodología de la Investigación 5ª Edición. México: International Thomson.

Fernández, M. A. (2015). Aplicación de Técnicas de Pruebas Automáticas Basadas en Propiedades a los Diferentes Niveles de Pruebas del Software. Trabajo Especial de Grado presentado ante la Universidad De Coruña, para optar al Título de Grado de Doctor. Coruña.

FondoNorma-ISO 9000:2006 Sistemas de gestión de la calidad. Fundamentos y Vocabulario (3ra Revisión).

Gacitúa, R. (2003). Métodos de Desarrollo de Software; El Desafío Pendiente de la Estandarización. Revista Teórica, 2003 Vol. 12, p. 23-42.

García, J., Moreira, A., Rossi, G. (2004). UML: el lenguaje estándar para el modelado del software. Disponible en: https://dialnet.unirioja.es/servlet/articulo?codigo=2064344

Gido, J. y Clements, J. (2008). Administración exitosa de Proyectos (3ra ed.). México: Cengage Learning.

Gonzalez, P. (2009). Generating Funtional Testing Case Method in Software Development. Revista Ingenierías Universidad de Medellín, 29-36.

Booch. G, Rumbaugh. J, Jacobson, I. The Unified Modeling Language User Manual. Addision-Wesley, 1999.

Gutierrez, J. (2011). Generació de Pruebas del Sistema a Partir de la Especificación Funcional. Trabajo Especial de Grado presentado ante la Universidad de Sevilla, para optar al Título de Grado de Doctor. Sevilla.

Hackney, J. (1992). Control y Gestión de Proyectos de Capital, segunda

Ed. MacGraw Hill, Inc. NEWYORK.

Hernández, R., Fernández, C. y Baptista, P. (2010) Metodología de la investigación (5ª Ed.). México, D.F., México: McGraw Hill Interamericana.

Humphrey, W. (2005). Practices: Acquiring Quality Software. Disponible en: http://www.worldebooklibrary.org/eBooks/WPLBN0002112533-Crosstalk--The-Journal-of-Defense-Software-Engineering--December-2005--Volume18--Issue-12---T-by-Johnstun--Kase.aspx

Humphrey, W. (2008). Software Quality: The Software Quality Challenge Disponible en: HYPERLINK "http://www.worldebooklibrary.org/eBooks/WPLBN0002112533-Crosstalk--The-Journal-of-Defense-Software-Engineering--June-2008--Volume-21-Issue-8---T-by-Johnstun--Kase.aspx" http://www.worldebooklibrary.org/eBooks/WPLBN0002112533-Crosstalk--The-Journal-of-Defense-Software-Engineering--June-2008--Volume-21-Issue-8---T-by-Johnstun--Kase.aspx

Hurtado J. (2010). EL proyecto de investigación (6ª ed.). Bogotá - Caracas: Ediciones Quirón.

IEEE. (2004) Guide to the Software Engineering Body of Knowledge. Disponible en: <ieeexplore.ieee.org/ielE/4425811/4425812/04425813.pdf

Jústiz, N.,Gómez, S., & Delgado, D. (2013). Testing process for software products at a quality laboratory. 131-145.

Ley de Reforma Parcial de la Ley Orgánica de Ciencia, Tecnología e Innovación (Gaceta Oficial No. 39.575 de fecha 16/12/2010).

Ley del sistema Venezolano para la Calidad (Gaceta oficial N° 37.543, de fecha 07/10/2002).

Lledó P. y Rivarola G. (2007). Gestión de Proyectos. Argentina: Prentice Hall y Pearson Educación.

Manifiesto for Agile Software Development (2014). Disponible en: http://agilemanifesto.org/

Montilva J. y Barrios J. (2007). Desarrollo de Software Empresarial. Universidad de los Andes.

Navarro, A., Fernández J. y Morales J. (2013). Revisión de metodologías ágiles para el desarrollo de software. Disponible en: http://dialnet.unirioja.es/servlet/articulo?codigo=4752083

Noriega, F. (2013). La Gerencia de Proyectos. Project Management, 50-62.

PDVSA (1997). Guías de Gerencia para Proyectos de Inversión de Capital (1997), Petróleos de Venezuela, Sociedad Anónima.

Palacio, L. (2009). Generating Funtional Testing Case Method in Software Development. 5-25.

Palella, S. Martins, F. (2006). Metodología de la Investigación Cuantitativa. Caracas: FEDUPEL.

Piattini, M. (2009) Calidad de Sistemas de Información, 2da ed., Madrid, RA-MA.

Porter, M. (1985). Competitive Advantage: Creating and Sustaining Superior Performance(1ra. Ed.). Nueva York: The Free Press

Pressman, R. (2010). Ingeniería del software. Un enfoque práctico (7ª ed.). España: McGraw-Hill.

Project Management Institute, Inc. (2013). A guide to the Project Management Body of knowledge. Fifth Edition. Newton Square: PMI

Project Management Institute, Inc. (2013). Código de Ética y Conducta Profesional. Disponible en:

https://www.pmi.org/~/media/PDF/Ethics/ap_pmicodeofethics_SPA-Final.ashx

Bien, A. (2013). Pruebas de Estress del Software. Disponible en: HYPERLINK "http://searchdatacenter.techtarget.com/es/cronica/Las-pruebas-de-estres-de-software-protegen-las-aplicaciones-empresariales-en-produccion" http://searchdatacenter.techtarget.com/es/cronica/Las-pruebas-de-estres-de-software-protegen-las-aplicaciones-empresariales-en-produccion ,

Vietyes, R (2004). Metodología de la Investigación en Organizaciones, Mercado y Sociedad: epistemología y técnicas (1 ra ed.). Buenos Aires: Editorial de las ciencias

Segura, S. (2010). Functional And Perfomance Testing of Feature Model Analysis Tools. Trabajo Especial de Grado presentado ante la Universidad de Sevilla, para optar al Título de Doctor. Sevilla.

Sommerville, I. (2005). Ingeniería de Software (7ma ed.). Mexico: Pearson Educación

"The case for Front End Loading: FEL and Constructability Reviews". Jones, Milton H., Professional paper delivered to the Greater New Orleans chapter, PMI. Disponible en: www.pmccinc.com/images/FrontEndLoading200409.pdf.

Valarino, E., Yáber, G. y Cemborain, M. (2011). Metodología de la investigación: paso a paso. México: Editorial Trillas.

ANEXOS

ANEXOS A: Contrato de Licencia de Uso y Garantía de Programas Contrato de Plan de Asistencia y Servicios

Entre **Empresa prestadora del SOFTWARE** Y **EL CLIENTE** más adelante identificados, se ha resuelto suscribir el presente contrato de Contrato de Licencia de Uso, Garantía Limitada de Programas y Plan Soporte y Servicios, el cual estará regido por los siguientes términos:

PRIMERA: DEFINICIONES: En el presente contrato, los términos utilizados tienen los siguientes significados:

a) **Empresa prestadora del SOFTWARE** es titular de los derechos de autor, conexos y similares, sobre los programas de computación y Software diseñado.

b) **LICENCIA DE USO**, tal como se usa en este contrato es el permiso que otorga **Empresa prestadora del Software** al **CLIENTE** para utilizar **LOS PROGRAMAS** identificados en el pago de la misma relacionada con este contrato.

c) **EL PROGRAMA** o **LOS PROGRAMAS**, tal como su usan en este contrato, significan los soportes lógicos de los computadores o software desarrollados por **Empresa prestadora del Software**, o cuyos derechos lo han cedido o licenciado sus autores.

d) **PROGRAMA MONOUSUARIO** O **COMPUTADOR INDIVIDUAL**, también conocido como "*stand alone*", significa que tanto la aplicación como la base de datos están instaladas en un computador y solo puede ser ejecutada en el computador donde haya sido instalada. Admiten múltiples contribuyentes, compañías o empresas.

e) **PROGRAMA MULTIUSUARIO BAJO REDES 1-4 COMPUTADORES**: Bajo esta modalidad la base de datos puede estar instalada en un servidor y ser ejecutada bajo redes hasta por 4 computadores definidos previamente por la empresa. Bajo este esquema los **PROGRAMAS** funcionan bajo la base de datos

MICROSOFT SQLSERVER EXPRESS de libre distribución aunque se recomienda dependiendo del volumen de datos y transacciones utilizar la base de datos **MICROSOFT SQLSERVER STANDARD**. Admiten múltiples contribuyentes, compañías o empresas.

f) **PROGRAMA MULTIUSUARIO BAJO REDES SIN LÍMITE DE COMPUTADORES**: Bajo esta modalidad la aplicación puede ser ejecutada bajo redes por tantos computadores como requiera la empresa. Bajo este esquema los PROGRAMAS funcionan bajo la base de **MICROSOFT SQLSERVER STANDARD**. Admiten múltiples contribuyentes, compañías o empresas.

g) EL CLIENTE es la persona natural o jurídica identificada en el pago emitido a **Empresa prestadora del SOFTWARE** que soporta la compra de la **LICENCIA DE USO** de **LOS PROGRAMAS**.

h) **EL USUARIO**, es cualquier persona natural que utilice LOS PROGRAMAS bajo la autorización o consentimiento **DEL CLIENTE**.

i) **LAS MARCAS**, tal como se usa en este contrato, significan los signos distintivos de productos o servicios, de la exclusiva propiedad de **Empresa prestadora del SOFTWARE**, y han sido solicitadas para registro en la República Bolivariana de Venezuela, o son usadas por **Empresa prestadora del SOFTWARE**, mediante autorización de su titular.

j) **LA GARANTIA LIMITADA**, tal y como se usa en el presente contrato, significa las obligaciones especiales que asume **Empresa prestadora del SOFTWARE** frente al **CLIENTE** de conformidad con las cláusulas de este contrato.

k) **FACTURA**, tal y como se usa en el presente contrato, significa el documento emitido por **Empresa prestadora del SOFTWARE** de acuerdo a la normativa legal vigente por las leyes de la República Bolivariana de Venezuela para la emisión de facturas, donde consten **LOS PROGRAMAS** adquiridos por **EL CLIENTE**.

l) **PLAN DE SOPORTE Y SERVICIO**: Es el servicio mediante el cual se extiende por un (1) año la validez de la cláusula QUINTA del presente contrato, siempre y cuando **EL CLIENTE** cancele el precio a tal efecto determine **Empresa prestadora del SOFTWARE** para este servicio.

SEGUNDA: OBJETO DEL CONTRATO:

La finalidad de este contrato que suscribe **Empresa prestadora del SOFTWARE** y **EL CLIENTE** es determinar las responsabilidades que asumen tanto **Empresa prestadora del SOFTWARE** como **CLIENTE** para utilizar **LOS PROGRAMAS** objeto del presente contrato.

TERCERA: **EL PRECIO**

El precio pagado por el producto objeto del presente contrato comprende para **EL CLIENTE:**

1. El derecho, no exclusivo que **Empresa prestadora del SOFTWARE** otorga a **EL CLIENTE** por medio del presente contrato a utilizar **EL** O **LOS PROGRAMAS** en un computador personal o en un servidor de redes.
2. La compraventa del empaque, CD O DVD que quedarán en poder de **EL CLIENTE**.
3. La compraventa del material impreso que acompaña al producto; el cual quedará en poder de **EL CLIENTE** sin que ello implique cesión de derechos intelectuales.

CUARTA: DURACION Y TERMINACION

Este contrato entrará en vigencia desde la fecha de la adquisición que se indique en el recibo de pago y continuará vigente por tiempo indefinido, con excepción de aquellas cláusulas en donde se limiten algunas garantías por un lapso definido de tiempo. Sin perjuicio de lo estipulado en el párrafo anterior, el término para reposición de la **LICENCIA DE USO** de **LOS PROGRAMAS** con cargo a **Empresa prestadora del SOFTWARE**, estará vigente hasta la fecha de vencimiento de la garantía. Las partes pueden dar por terminado el presente contrato por mutuo

consentimiento o mediante aviso escrito enviado por correo electrónico a la direcciones registradas al momento de contratar por con lo menos treinta (30) días, de anticipación a la fecha en que este contrato o alguna de sus cláusulas deba ser terminado

QUINTA: GARANTIAS Y LIMITACIONES

Empresa prestadora del SOFTWARE garantiza por un período de un (1) año contado a partir de la emisión de la factura que soporta la adquisición de la **LICENCIA DE USO** de **LOS PROGRAMAS** o del **PLAN DE ASISTENCIA Y SERVICIOS que:**

1. Suministrará al **CLIENTE** atención telefónica o vía e-mail en forma limitada para aclarar dudas en la utilización del **PROGRAMA**.

2. Publicará en la página Web www.XXXX información sobre las correcciones, cambios y nuevas versiones que se produzcan en **LOS PROGRAMAS** y notificará al **CLIENTE** enviando un correo electrónico a la dirección que indique al momento de contratar. **Empresa prestadora del SOFTWARE** no será responsable en caso que por cualquier causa dicho correo electrónico no haya podido ser entregado.

3. Pondrá a disposición de **EL CLIENTE** en la página Web www.XXXX todas las mejoras que se produzcan en **LOS PROGRAMAS** adquiridos, este servicio no tendrá costo siempre y cuando el **CLIENTE** lo obtenga vía Internet. **Empresa prestadora del SOFTWARE** podrá suministrar las nuevas versiones durante el lapso especificado por otros medios y en este caso el **CLIENTE** se compromete a cancelar el precio que establezca **Empresa prestadora del SOFTWARE** por este servicio, más los gastos de envío o el servicio de instalación computado en horas técnicas de servicio según sea el caso. **EL CLIENTE** cancelará estos servicios, de contado, en la oportunidad de su prestación. Cualquier servicio técnico asociado con el mantenimiento de este programa será con costo a cargo de **EL CLIENTE**. **Empresa prestadora del SOFTWARE** facturará estos servicios en la oportunidad en que se produzcan. Para obtener las mejoras de **LOS PROGRAMAS**, **EL CLIENTE** es responsable de:

a. Comunicarse vía internet con el servicio en la página Web www.XXXX

b. Copiar las nuevas versiones de los **PROGRAMAS** y

c. Solicitar a **Empresa prestadora del SOFTWARE**, en caso de ser necesario, las claves necesarias para activar las nuevas versiones de sus **PROGRAMAS**.

SEXTA: EXCEPCIONES

Con excepción de lo señalado anteriormente, **Empresa prestadora del SOFTWARE**, no ofrece garantías explícitas o implícitas, ni asume responsabilidad alguna por cualquier daño o perjuicio, de cualquier índole, que pueda sufrir **EL CLIENTE**, incluyendo daños y perjuicios o imposibilidad de utilizar este **PROGRAMA**. **Empresa prestadora del SOFTWARE** no será responsable por los daños y perjuicios ocasionados a **EL CLIENTE** a terceras personas no autorizadas en virtud de las infracciones en contra de sus derechos patrimoniales sobre el **PROGRAMA** y por la contravención de las estipulaciones del contrato de licencia. Asimismo **Empresa prestadora del SOFTWARE** no asume responsabilidad alguna y así lo reconoce explícitamente y solidariamente **EL CLIENTE**, por las infracciones, delitos, infracciones y delitos tributarios que pudieran cometer **EL CLIENTE** y/o **SUS USUARIOS** mediante el uso de este programa, en grado de autor o coautor. Esta declaración también se hace extensiva a los cómplices o encubridores de **EL CLIENTE** en las mencionadas infracciones o delitos. **EL CLIENTE** y **Empresa prestadora del SOFTWARE**, manifiestan que conocen que este producto no está normalizado por entidades públicas gubernamentales o no gubernamentales y que en todo caso, la responsabilidad de **Empresa prestadora del SOFTWARE**, sólo asciende al monto del valor pagado por **CLIENTE** al adquirir el programa. En la máxima medida permitida por la legislación aplicable, **Empresa prestadora del SOFTWARE** renuncia a todas las demás garantías, expresas o tácitas incluyendo entre otras, garantías implícitas de comercio e idoneidad para un determinado fin con respecto al programa de computación, así como de los materiales escritos adjuntos al mismo. La presente garantía limitada le concede a **EL CLIENTE** derechos legales específicos y podrán competirle otros

que varían según el estado o territorio. En la medida máxima permitida por la legislación aplicable, **Empresa prestadora del SOFTWARE** no se responsabilizarán de daños (incluyendo entre otros, daños directos o indirectos por lesiones a las personas, lucro cesante, interrupción de actividad comercial, pérdida de información comercial o cualquier otra pérdida pecuniaria) que derive del uso o incapacidad de usar el programa de computación, incluso si **Empresa prestadora del SOFTWARE** ha sido informado de la posibilidad de dichos daños. En cualquier caso, toda la responsabilidad de **Empresa prestadora del SOFTWARE** en virtud de cualquier estipulación de este contrato de licencia se limitará a la cantidad efectivamente pagada por **EL CLIENTE** por la licencia del **PROGRAMA**. De no acatar los lineamientos estipulados el riesgo total en cuanto al rendimiento del programa de computación, en caso de estar defectuoso, **EL CLIENTE** (y no **Empresa prestadora del SOFTWARE**) asumirá el costo total de los servicios, reparaciones o correcciones necesarios. **Empresa prestadora del SOFTWARE** no garantiza que las funciones contenidas en el programa de computación pueden satisfacer sus requisitos o que la operación del programa de computación sea de forma ininterrumpida o que esté libre de errores, o que los defectos del programa de computación serán corregidos.

SEPTIMA: COMPROMISOS DEL CLIENTE

EL CLIENTE se compromete a:

1. "Registrarse como usuario en www.XXXX"a dentro de los treinta (30) días continuos siguientes a la fecha de compra indicada en la factura de adquisición de la **LICENCIA DE USO** de **LOS PROGRAMAS**.

2. Revisar periódicamente la página Web www.XXXX para estar informado sobre cambios y mejoras que se produzcan en **LOS PROGRAMAS**.

3. Mantener actualizados sus datos en www.XXXX para que sea posible enviarle vía correo electrónico información actualizada.

4. Revisar detalladamente los resultados de los cálculos que realicen **LOS PROGRAMAS** e informar inmediatamente a **Empresa prestadora del SOFTWARE** y a terceros que pudieran estar afectados sobre cualquier diferencia en cálculo, por criterios legales o de cualquier otra naturaleza. **EL CLIENTE** es el único responsable ante terceros por las consecuencias que pudiera tener la utilización de los resultados de **LOS PROGRAMAS** ya que su responsabilidad revisarlos y validarlos antes de utilizarlos.

5. No permitir, ni efectuar copias de este programa o del manual de instrucciones, o del material adicional recibido.

6. No ceder, expresa o tácitamente, el presente contrato, o la **LICENCIA DE USO** de **LOS PROGRAMAS** o impresos que a se refieren; a no usar las MARCAS de los productos o servicios relacionados con el contrato; siendo entendido que cualquier violación a lo anterior, dará lugar al ejercicio de las acciones penales, civiles o administrativas que consagre la ley en favor de **Empresa prestadora del SOFTWARE**.

7. Pagar cualquier servicio técnico asociado con el mantenimiento de **LOS PROGRAMAS**. **Empresa prestadora del SOFTWARE** facturará estos servicios en la oportunidad en que se produzcan.

8. Mantener en sus equipos licencias originales de los Sistemas Operativos, los cuales deben haber sido adquiridos legalmente por **EL CLIENTE** sin que **Empresa prestadora del SOFTWARE**, asuma responsabilidad alguna por el funcionamiento de dicho sistema operativo ya bien sea en computadores personales ni en Redes de computadores.

9. No permitir copias del programa o del material impreso que se acompaña, siendo solidariamente responsable por los daños y perjuicios que dicha conducta pueda ocasionar, y por cualquier clase de reclamos judiciales o extrajudiciales y daños que puedan llegar como resultado de la copia no autorizada del programa o del material impreso que los acompaña, o del empaque del mismo, o de la reventa no autorizada, y reembolsará a **Empresa prestadora del SOFTWARE**, todas las

sumas que esta sociedad haya tenido que pagar en relación con dichas reclamaciones o perjuicios, incluyendo honorarios de abogado y costa del proceso o procesos, si hubiere lugar a ello.

El CLIENTE declara conocer el alcance de los términos y definiciones establecidos en la cláusula **PRIMERA** del presente contrato y manifiesta expresamente que al instalar, copiar, solicitar la activación permanente y/o de otra forma usar la **LICENCIA DE USO** de **LOS PROGRAMAS** está de acuerdo en quedar obligado por las cláusulas de este contrato. En caso de no estar de acuerdo con los términos del presente contrato, **EL CLIENTE** no puede usar o copiar el programa de computación, y debe rápidamente ponerse en contacto con **Empresa prestadora del SOFTWARE** para obtener instrucciones sobre la devolución de los productos no utilizados y el reembolso del importe pagado.

OCTAVA: El CLIENTE puede solicitar el reintegro del dinero cancelado en la factura que soporta la adquisición del **PROGRAMA** dentro de los treinta (30) días calendario siguientes a su adquisición siempre y cuando manifieste por escrito su intención, devuelva a **Empresa prestadora del SOFTWARE** el original de la factura y demuestre a satisfacción de **Empresa prestadora del SOFTWARE** que LOS **PROGRAMAS** han sido desinstalados de sus computadores. En caso afirmativo **Empresa prestadora del SOFTWARE** procederá al reintegro del precio pagado por **LOS PROGRAMAS** dentro de los treinta (30) días calendario siguientes a la recepción de la notificación

NOVENA: El presente contrato está regido por la Leyes de la República Bolivariana de Venezuela y no podrá ser modificado por terceros, a menos que medie comunicación escrita, suscrita por el representante legal de **Empresa prestadora del SOFTWARE**.

En Testimonio de lo anterior, **EL CLIENTE** firma este contrato en la Ciudad de Caracas en cuyos tribunales se domicilia el presente contrato.

Anexo B Formato de Reporte de No Conformidad

Nro No Conformidad:		Cliente:	
Area Proceso:			
Origen de la No Conformidad			
☐ Reclamo		☐ Auditoria	☐ Otros (Especificar)
Sección I: Detalle de la No Conformidad			
Descripción:			
Analisis de las Causas:			
Sección II Plan de Acción Propuesto			
Acción Adopatada	☐ Correctiva	☐ Preventiva	
Responsable Nombre y Apellido, Fecha,Firma			
Responsable de la Implantación:		Plazo para la Implantación:	Fecha para Control y Seguimiento:
Sección III Seguimiento y Control			
Comprobación de la Implantación:		Comprobación de la Eficacia:	
☐ Ejecutada	☐ No Ejecutada	☐ Aceptable ☐ No Aceptable	☐ Pendiente
Observaciones:			
Responsable Nombre y Apellido, Fecha,Firma			

Anexo C Formato del Instrumento de Validación de Usabilidad

Formato del Instrumento de Validación de Usabilidad

Este instrumento ha sido creado con la finalidad de llevar un registro de las opiniones de los usuarios finales hacerca de la usabilidad del sistema, es decir, las interfaces del software lo que incluye, el tamaño de la letra, los colores utilizados en las pantallas, la navegación apropiada en cada uno de los módulos, entre otros, la información proporcionada por los actores de las pruebas será de gran ayuda para mejora el software en cuanto a los problemas de usabilidad.

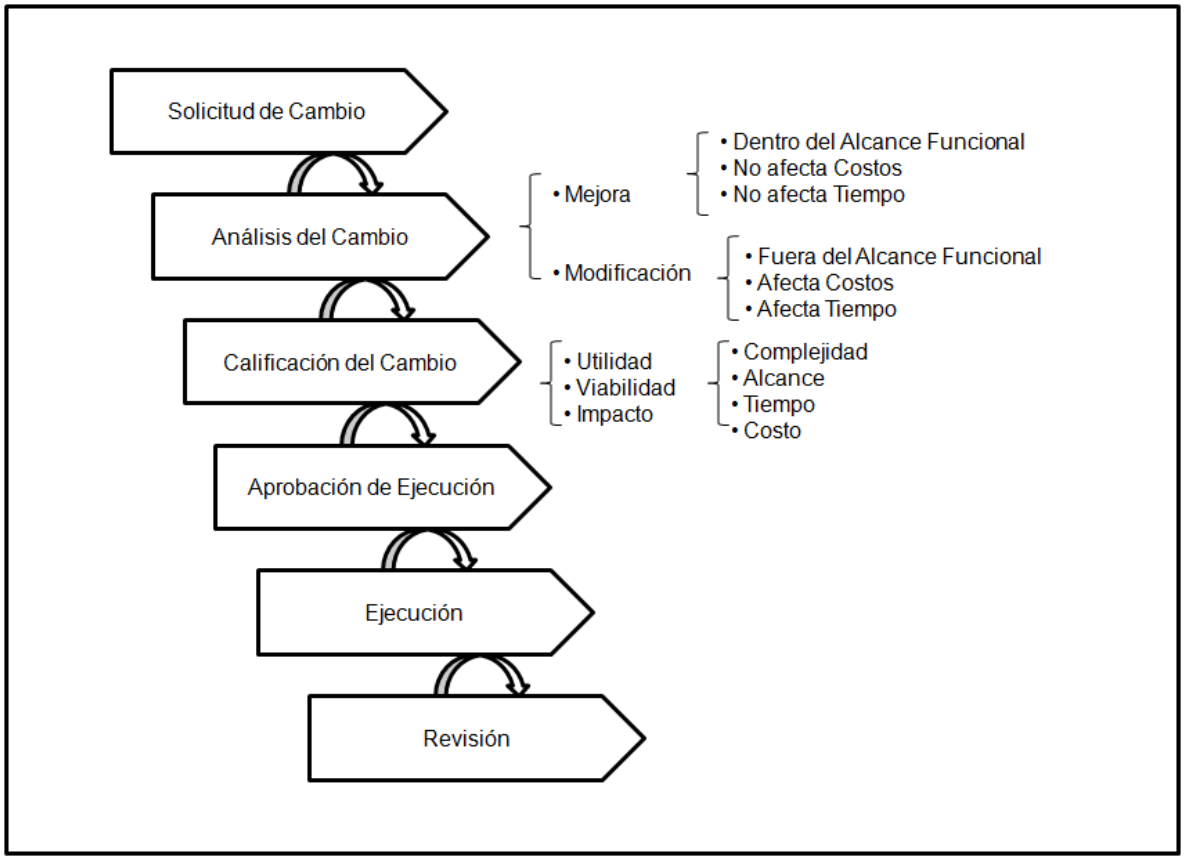
Nombre	Descripción	Fecha Aplicación

Seleccione la opción que mas se acerque a su opinión

ASPECTOS A EVALUAR EN LA APLICACIÓN	VALORACION SI	MED	NO
Incluye información relevante, necesaria y sencilla.			
Utiliza frases, palabras y conceptos familiares.			
Sigue convenciones del mundo real.			
Usa objetos, acciones y/o opciones para evitar que el usuario tenga que recordar información. Ejemplo menu desplegables.			
Provee objetos visibles e intuitivos			
La composición de las pantallas son las mismas en toda la aplicación. Cada control tiene asignado una única función y siempre es la misma.			
Siempre informa acerca de lo que está ocurriendo.			
Permite deshacer y rehacer acciones realizadas.			
Incluye acciones para hacer mas rapida la interacción.			
Incluye mensajes de error expresados en un lenguaje comun y sencillo, indicando el problema y sugeriendo soluciones de forma constructiva.			
Incluye acciones para prevenir la existencia de errores.			
Incluye ayuda para asistir al usuario.			
Existe legibilidad y claridad textual (tipo, color y tamaño adecuado de la fuente utilizada)			
Existe organización adecuada del contenido y los elementos.			

Nombre del Testing	Observaciones	Firma

Anexo D Procedimiento De Control De Cambio



Se realiza la solicitud del cambio al equipo encargado de analizar el cambio, el mismo lo analiza he indica si procede una mejora, o si por el contrario es una modificación al requerimiento, luego se indica si califica para el cambio, se aprueba la ejecución del mismo, se realizan los cambios y posterior se realiza una revisión del software cumpliendo con pruebas de regresión al módulo o proceso programado, si cumple con lo requerido, con la necesidad que solicito el analista de pruebas.

Anexo E: Flujo de Proceso de Pruebas