PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DISERTACIÓN PREVIA A LA OBTENCIÓN DEL TÍTULO DE

INGENIERO EN SISTEMAS Y COMPUTACIÓN

"ANÁLISIS COMPARATIVO DE LOS MÉTODOS DE CAJA NEGRA, BLANCA Y GRIS DE TESTING DE SOFTWARE, PARA LA PROPUESTA DE UN MÉTODO ESTÁNDAR DE TESTING"

AUTOR: Tulio Navarrete Vásquez

DIRECTORA: Ing. Beatriz Campos

QUITO, 2012

DEDICATORIA

En otra época quizás esta dedicatoria hubiese sido otra, pero hoy, este paso está dedicado a ese motorcito incansable, lleno de energía y alegría, esto y lo que venga es para Tulio Nicolás, mi hijo, también a mi compañera un apoyo incondicional en mi vida, María Fernanda, mi amada esposa.

AGRADECIMIENTOS

A Dios y a mi ángel La Dolorosa Del Colegio, a mi Madre por su empeño, esfuerzo y paciencia, a mis abuelitos Arturo y Adriana por todo lo hecho por mí desde muy niño.

Infinitas Gracias!

TABLA DE CONTENIDOS

1 INTRODUCCIÓN AL PROCESO DE TESTING DE SOFTWARE 1

1.1 Antecedentes 1

1.2 Ciclo de vida de software 2

1.2.1 Modelo en cascada 2

1.2.2 Modelo de prototipos 3

1.2.3 Modelo del ciclo de vida en espiral 5

1.2.4 Modelo V 6

1.2.5 TestingMaturityModel 7

1.2.6 Test ProcessImprovement 9

1.3 Etapa de testing en el ciclo de vida del software 9

1.3.1 Aseguramiento de la calidad del software 10

1.4 Niveles de pruebas 11

1.4.1 Requerimientos 12

1.4.2 Análisis 14

1.4.3 Diseño 14

1.4.4 Implementación 15

1.4.5 Pruebas 15

1.4.6 Utilidad 19

1.4.7 Confiabilidad 20

1.4.8 Robustez 20

1.4.9 Desempeño 20

1.4.10 Corrección 21

1.5 Métricas y medición 21

2 PROCESO DE TESTING 23

2.1 MODELOS DEL PROCESO DE TESTING 23

2.2 PLANEAMIENTO Y CONTROL 24

2.2.1 Planeamiento 24

2.2.2 Controles 27

2.2.3 ISO/IEC 9126 27

2.3 ANÁLISIS COMPARATIVO DE LOS PRINCIPALES MÉTODOS DE TESTING DE SOFTWARE 28

2.3.1 Prueba Estructural 29
2.3.2 Prueba Funcional: 29
2.3.3 Prueba de integración: 30
2.3.4 Prueba de Aceptación 30
2.3.5 Prueba del sistema 30
2.3.6 Pruebas de Regresión: 31
2.3.7 Prueba unitaria: 31
2.3.8 Prueba de Humo: 32
2.4 IMPLEMENTACIÓN Y EJECUCIÓN 34
2.5 EVALUACIÓN Y REPORTE 36
2.6 CLAUSURA DE LAS ACTIVIDADES 37
3 DOCUMENTACIÓN DEL PROCESO TESTING 39
3.1 ESPECIFICACIÓN DE REQUERIMIENTOS 39
3.1.1 Requerimientos funcionales 40
3.1.2 Requerimientos no funcionales 40
3.2 DOCUMENTO DE LA ESTRATEGIA DE TESTING 42
3.3 ESPECIFICACIÓN DE INTERFACES 43
3.4 TEST PLAN 44
3.5 DOCUMENTO DE MONITOREO DE TESTING 45
3.6 ESPECIFICACIÓN DE CASOS DE PRUEBA 46
3.7 ESPECIFICACIÓN DE PROCESOS DE PRUEBA 46
3.8 REPORTE DIARIO DE PRUEBAS 47
3.9 REPORTE DE ERRORES O BITÁCORA DE INCIDENCIAS 48
4 HERRAMIENTAS DE TESTING 50
4.1 INTRODUCCIÓN 50
4.2 HERRAMIENTAS AUTOMÁTICAS DE TESTING 51
4.3 HERRAMIENTAS DE ANÁLISIS DINÁMICO 52
4.4 HERRAMIENTAS DE MEDICIÓN 52
4.5 HERRAMIENTAS DE RENDIMIENTO DE TESTING 52
4.6 HERRAMIENTAS DE COMPRENSIÓN Y ANÁLISIS DE CÓDIGO FUENTE 53
4.7 HERRAMIENTAS DE ANÁLISIS ESTÁTICO 54
5 ANÁLISIS DE REQUERIMIENTOS PARA EL TESTING DE SOFTWARE 58
5.1 INTRODUCCIÓN 58

5.2 REQUERIMIENTOS FUNCIONALES 60
5.3 REQUERIMIENTOS NO FUNCIONALES 61
5.3.1 Escalabilidad 62
5.3.2 Disponibilidad 62
5.3.3 Seguridad 63
5.3.4 Mantenibilidad 63
5.4 REQUERIMIENTOS TÉCNICOS 65
5.5 REQUERIMIENTOS LEGALES 66
6 EXTRACCIÓN 69
6.1 ANÁLISIS DE LAS PRINCIPALES CARACTERÍSTICAS DE MÉTODOS TESTING 69
6.2 METODO DE CAJA NEGRA 70
6.3 METODO DE CAJA BLANCA 72
6.4 METODO DE PRUEBAS DE INTEGRACION 73
6.5 METODO DE PRUEBAS DE ACEPTACION 75
7 ESTUDIO DE MERCADO 77
7.1 DISEÑO DE LA ENCUESTA 77
7.2 RECOLECCION DE DATOS 80
7.3 PROCESAMIENTO DE DATOS 80
7.4 ANALISIS E INTERPRETACION DE DATOS 80
8 FORMULACION DE UN NUEVO METODO DE TESTING 106
8.1 Planificación Control de Versionamientos 107
8.2 Planificación Control de cambios. 107
8.3 Simulación de ambientes: 108
8.3.1 Ventajas: 109
8.3.2 Personal Capacitado: 110
8.3.3 Diagrama de Flujo de pruebas con el método propuesto. 110
9 PRUEBA DE TESTING DE SOFTWARE APLICANDO EL METODO PROPUESTO 112
9.1 DEMOSTRACION DEL MODELO 112
10 CONCLUSIONES Y RECOMENDACIONES 118
10.1 CONCLUSIONES 118
10.2 RECOMENDACIONES 119
11 Bibliografía 121
12 Anexos 123

Capítulo I

1 INTRODUCCIÓN AL PROCESO DE TESTING DE SOFTWARE

1.1 Antecedentes

Desde el punto de vista del cliente y los usuarios, la calidad de un producto de software es percibida principalmente por las fallas que encuentran en el producto y por la gravedad que éstas tienen para el negocio del cliente. Para ser competitivas, las empresas desarrolladoras de software necesitan asegurarse de la calidad de sus productos previo a su instalación en el ambiente del cliente. La prueba exhaustiva del producto antes de ser entregado al cliente implica probar el comportamiento del mismo para todas las combinaciones válidas e inválidas de entradas, bajo cada estado posible del sistema. Esto, incluso para un programa pequeño puede llevar cientos de años y es económicamente inviable. Debido a esto, las empresas que desarrollan software intentan una solución de compromiso, esto es entregar sus productos con la menor cantidad de defectos posible. Para esto, definen cuando entregar el producto a los usuarios en función del costo y el beneficio de realizar las pruebas.

El desafío está en encontrar los defectos que tendrían mayor impacto negativo para el negocio del cliente antes de que el producto se haya entregado. Para lograr este objetivo, existen técnicas y estrategias para definir y ejecutar las pruebas y es necesario contar con un equipo entrenado en ellas. Este equipo puede ser parte de

la organización de desarrollo o puede ser externo. Ambas formas tienen sus ventajas y desventajas.

1.2 Ciclo de vida de software

El proceso de desarrollo de software consta de varias tareas, pero existen varios métodos de aplicar estas tareas al proceso de desarrollo, entre los más comunes podemos mencionar al modelo en cascada, espiral, y modelo de prototipos.

Estas Tareas son:

- Análisis
- Diseño
- Construcción y pruebas
- Implantación
- Mantenimiento

Describiré a continuación las características más importantes de los modelos antes mencionados.

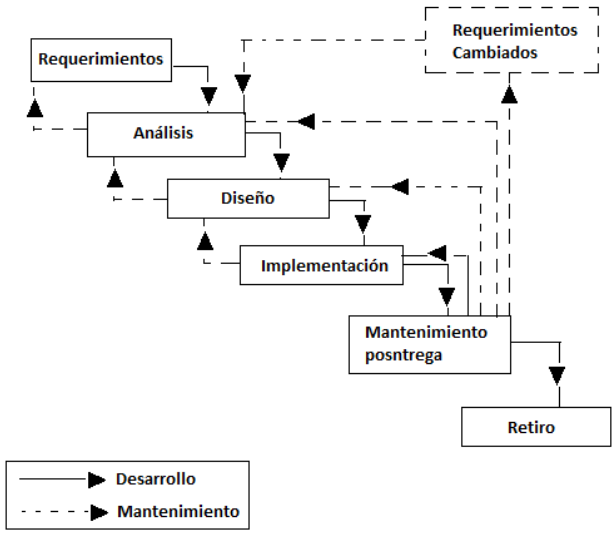
1.2.1 Modelo en cascada

En este modelo no se completa ninguna fase hasta que se haya terminado la documentación para esta fase, y los productos de esta fase hayan sido aprobados por el equipo de calidad.

Esto con lleva modificaciones sobre la marcha, si los productos de una fase previa tienen que cambiarse no se procederá a la siguiente hasta que la

documentación de esta fase se haya modificado y el equipo de calidad haya revisado y aprobado las modificaciones.

Figura 1: MODELO EN CASCADA[1]



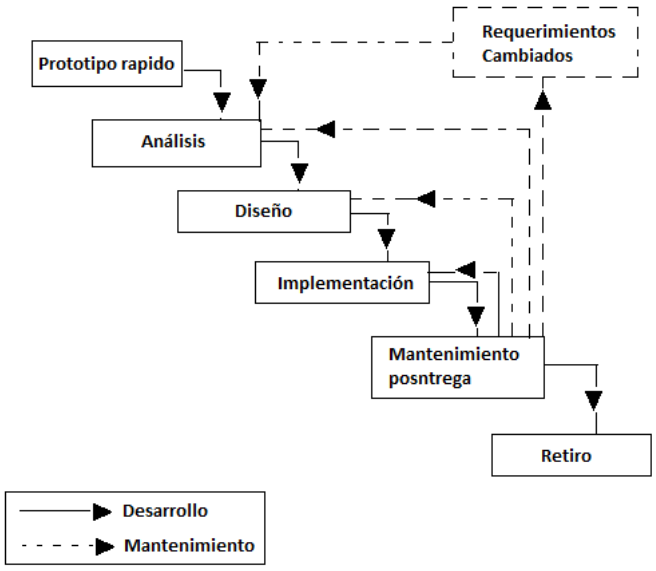
1.2.2 Modelo de prototipos

El modelo de prototipos o prototipos rápido es un modelo de trabajo cuya función es equivalente a un subconjunto del producto .Por ejemplo, si el producto va a manejar cuantas por pagar, cuentas por cobrar, y almacenaje, entonces el prototipo podría consistir en un producto que lleve a cabo el manejo a través de la pantalla para la captura de datos e imprima reportes, pero que no actualice archivos ni maneje errores.

[1] http://easydevelopmentfiis.blogspot.com/2009/12/software-development-methodologies-and.html

El primer paso de este modelo es construir un prototipo rápido y dejar que el cliente y los futuros usuarios interaccionen y experimenten con él. SI el cliente cree que el prototipo hace la mayor parte de lo que necesita, se pueden redactar un documento de especificaciones con la garantía de que el producto será conforme las necesidades reales del cliente.

Figura 2: MODELO EN PROTOTIPOS[2]



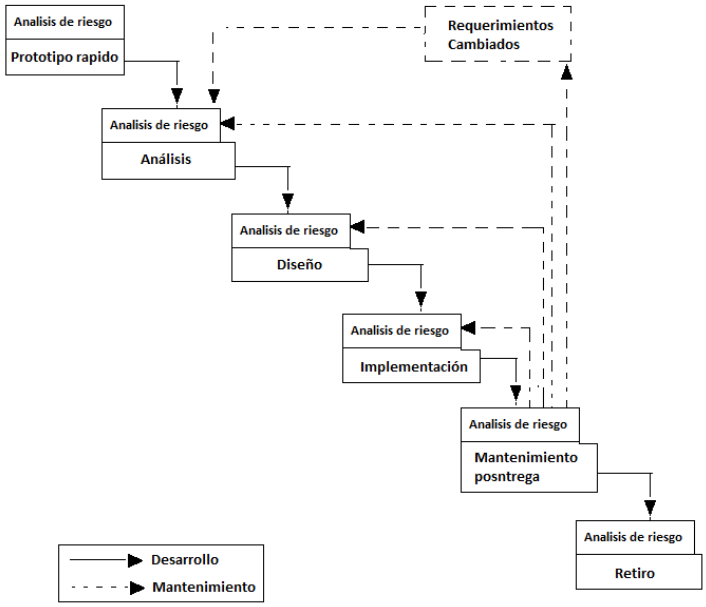
Debido a que el desarrollo de este modelo se basa en un prototipo real y probado es posible que se hagan menos necesarios los bucles de retro alimentación, ya que contamos con información real y no tan solo documentos de requerimientos, y esta quizás es la principal fortaleza del modelo.

[2] http://es.wikipedia.org/wiki/Modelo_de_prototipos

1.2.3 Modelo del ciclo de vida en espiral

La idea principal de este modelo es el de minimizar los riesgos que tiene un proyecto de software, usando prototipos y otros medios, este modelo es como un modelo en cascada pero cada fase precedida de un análisis de riesgo. Antes de comenzar cada fase, se hace un intento por reducir los riesgos y si es imposible controlar todos los riesgos significativos en esta etapa, entonces se termina inmediatamente el proyecto.

Figura 3: MODELO EN PROTOTIPOS[3]



[3] http://html.rincondelvago.com/modelos-de-procesos-de-software.html

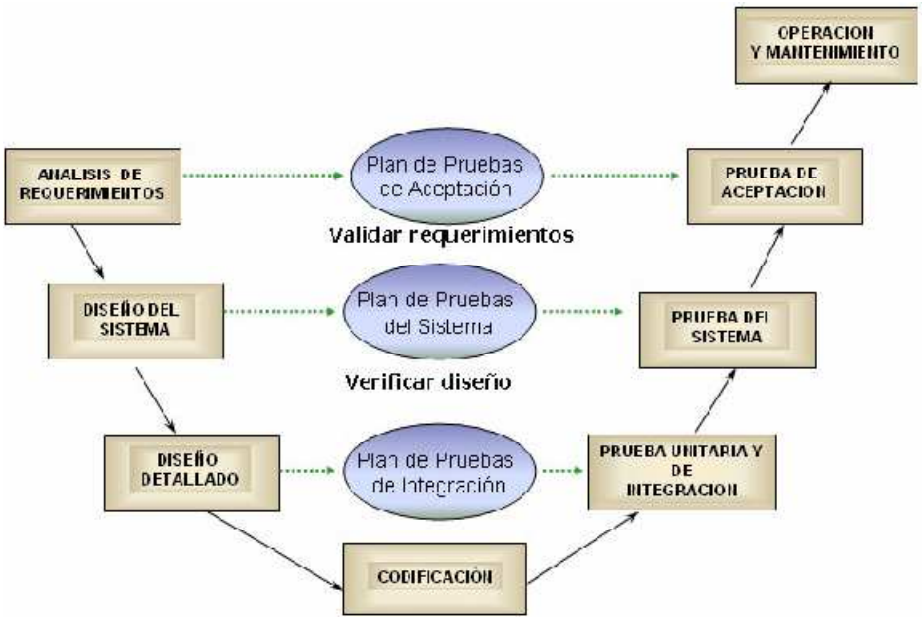
1.2.4 Modelo V

El modelo V es una variación del modelo en cascada que demuestra cómo se relacionan las actividades de prueba con las de análisis y diseño.

El modelo V sugiere que la prueba unitaria y de integración sea hecha también para verificar el diseño, verificando que todos los aspectos del diseño del programa se han implementado correctamente en el código, la prueba del sistema debe verificar el diseño del sistema, asegurando que todos los aspectos del diseño están correctamente implementados. La prueba de aceptación que es dirigida por el cliente, valida los requerimientos.

Figura 4: MODELO V[4]

[4]http://jummp.wordpress.com/2011/09/06/desarrollo-de-software-testing-modelo-en-v/



1.2.5 TestingMaturityModel

Es un complemento al CMM[5] y existe una correspondencia directa entre los niveles en ambos modelos. TMM contiene un conjunto de niveles para que la organización mejore la madurez de su proceso de prueba, un conjunto de prácticas recomendadas en cada nivel de madurez y un modelo de evaluación que permite a la organización evaluar y mejorar su proceso de prueba. Cada nivel de madurez representa una etapa en la evolución hacia un proceso de prueba maduro. Moverse de un nivel al siguiente implica que las prácticas del nivel anterior continúan realizándose.

[5] CapabilityMaturityModel : ayuda a las organizaciones a mejorar sus procesos de desarrollo en múltiples disciplinas además del software.

A continuación se detallan las características de cada nivel de TMM.

Nivel 1 – Inicial: El proceso de prueba es caótico. No hay objetivos de madurez planteados para este nivel.

Nivel 2 – Definido: La prueba está separado del debugging y es definido como una fase que sigue a la codificación. Es una actividad planificada. El objetivo primario de la prueba en este nivel es mostrar que el software cumple con sus especificaciones.

Nivel 3 – Integración: La prueba es parte del ciclo de vida del software. La planificación de la prueba comienza en la fase de Requerimientos y continúa a través del ciclo de vida, siguiendo una variación del Modelo V. Los objetivos de la prueba son establecidos con respecto a los requerimientos basados en las necesidades de cliente y usuarios y son usados para el diseño de las pruebas y los criterios de éxito.

Nivel 4 – Gerenciado y Medido: El proceso de prueba es medido y cuantificado. Las revisiones en todas las fases del proceso de desarrollo son reconocidas como actividades de prueba y control de calidad.

Nivel 5 – Optimización, Prevención de Defectos y Control de calidad: Existen mecanismos para la mejora continua de la Prueba. Se realiza prevención de defectos y control de calidad.

1.2.6 Test ProcessImprovement

Brinda guías prácticas para evaluar el nivel de madurez de la prueba en una organización y para mejorar paso a paso el proceso. Los componentes del modelo son:

- ✓ **Áreas Claves**: Permiten mejorar y estructurar el proceso de prueba. Existen 20 áreas claves.
- ✓ **Niveles**: La forma en que las áreas claves son organizadas en el proceso es lo que determina la madurez del proceso. Existen los niveles: A, B, C y D. Cada nivel mejora el anterior en términos de tiempo, dinero y calidad. Cada nivel consiste de ciertos requerimientos para las áreas claves, los niveles mayores comprenden los requerimientos de los niveles inferiores.
- ✓ **Checkpoints**: Los requerimientos para cada nivel están dados como checkpoints.
- ✓ **Matriz de Madurez del Test**: No todas las áreas claves son igualmente importantes, los niveles y las áreas claves están relacionadas en una matriz de madurez del test, mostrando las prioridades y dependencias entre los niveles y las áreas claves.

1.3 Etapa de testing en el ciclo de vida del software

Todos los modelos del ciclo de vida de software clásicos incluyen una fase de pruebas independiente, después de la integración y antes del mantenimiento post-

entrega, práctica que resulta caduca e ineficiente si lo que se quiere es conseguir un software de alta calidad. Las pruebas deben ser un componente integrado del proceso de programación y es una actividad que debe llevarse a cabo a lo largo de todo el ciclo de vida. Se debe verificar desde la etapa de requerimientos; cada uno de los componentes también se debe probar uno a uno, mientras se van desarrollando hasta llegar al producto en su totalidad. Al finalizar e integrar todos los componentes también se debe realizar pruebas de funcionalidad. Después de pasar las pruebas de aceptación, se instalará el producto para comenzar el proceso de mantenimiento post-entrega, a partir de esta etapa se realizaran pruebas de cada una de las versiones modificadas de los componentes y del producto final antes de poner en producción.

1.3.1 Aseguramiento de la calidad del software

La tarea de cada profesional de software es garantizar software de alta calidad, es decir cada uno de los involucrados en el desarrollo y mantenimiento es responsable de verificar que su trabajo sea correcto. La calidad no es algo que añada el grupo de control de calidad (SQA, por sus siglas en ingles), sino más bien debe ser construida por los desarrolladores desde el principio. Una función del grupo de SQA es garantizar que los desarrolladores estén haciendo en realidad un trabajo de alta calidad.

Más específicamente, una vez que los desarrolladores hayan terminado un flujo de trabajo y hayan revisado que este ha terminado correctamente, la función del equipo de SQA es garantizar que esto sea así. Y obviamente al final cuando el producto este completo y los desarrolladores estén seguros que el producto en su totalidad es correcto el grupo de SQA también debe asegurarse de eso.

SQA se enfoca en el proceso de software, así por ejemplo, entre las responsabilidades del grupo de SQA se encuentra desarrollar diversas normas a partir de las cuales el software se debe desarrollar, así como establecer procedimientos de supervisión para garantizar el cumplimiento de dichas normas. En resumen, la función de SQA es garantizar el cumplimiento del proceso de programación y con ello la calidad del producto.

1.4 Niveles de pruebas

Existen dos aspectos importantes para las pruebas:

a) Toda persona de desarrollo y mantenimiento es responsable de garantizar que su trabajo sea correcto.

b) Una vez que la persona de desarrollo o mantenimiento está convencida de que se su producto esta correcto, este será manejado por el grupo de control de calidad, para realizar pruebas integrales.

Los niveles de pruebas cambian dependiendo del componente que se va a probar, pero la principal característica de estos debe ser que provean la posibilidad de darles seguimiento. Como se ha dicho, toda etapa en el ciclo de vida de software debe proveer la posibilidad de darle seguimiento, teniendo en claro cuáles son estas etapas y no pensar en que una es más importante que otra, ya que para pasar de una a otra etapa el equipo de SQA deberá aprobarla.

A continuación se detalla cada etapa, especificando lo que realmente se debe probar en cada etapa.

1.4.1 Requerimientos

Los requerimientos describen el “QUE” de un sistema y no el “COMO” los requerimientos son un documento escrito en lenguaje natural y poco técnico y contiene una descripción de que hará el sistema mas no de cómo lo hará.

Tras la elaboración de este documento existe un proceso donde participa el cliente (quien necesita el sistema) ya que este es quien conoce las

necesidades que el sistema debe satisfacer. Después del análisis, y descripción del problema que desea solucionar con el sistema el cliente generará un documento de requerimientos, donde detallara sus necesidades y expectativas con respecto al sistema que desea como producto final, es importante que este documento sea revisado por el desarrollador y solicitar aclaraciones de ser necesarias de manera que se tenga claro lo que se debe desarrollar y además este documento debe dar la facilidad de hacerle un seguimiento durante el proceso de desarrollo.

Existen dos tipos de requerimientos, los requerimientos funcionales y los no funcionales; los funcionales definen factores tales como los parámetros de entrada y salida, formato, almacenamiento, estructura, capacidades de cálculo, calendario y sincronización.

Los no funcionales en cambio definen las propiedades o cualidades del producto como usabilidad, eficiencia, rendimiento, espacio, portabilidad, etc.

1.4.2 Análisis

Una vez que se tienen los requerimientos expuestos por el cliente en el documento de requerimientos el proceso de detallarlos y extenderlos se denomina análisis de requerimientos, además se tratara de pasar el documento de requerimientos a un lenguaje algo más técnico sin que este proceso altere la esencia de los que el cliente detallo en el documento anterior sino más bien con el fin de ponerlo en un lenguaje que sea más familiar para los equipos que formaran parte del desarrollo del proyecto en todas las etapas del ciclo de vida.

1.4.3 Diseño

Un aspecto fundamental de la ejecución de todas las etapas del ciclo de vida de software es el dar seguimiento, en el caso del diseño esto significa que cada parte que se diseñe debe estar ligada a una parte del análisis, un diseño bien desarrollado permitirá al equipo de desarrollo una herramienta poderosa para comprobar si el diseño coincide con las especificaciones.

En esta etapa por lo general el cliente no está presente ya que esta tiene una naturaleza técnica, y en esta etapa se debe contar con todos o algunos

miembros del grupo de calidad para garantizar que el diseño este correcto, estos buscaran fallas de la lógica, de la interface, manejo de excepciones y sobre todo la incongruencia con las especificaciones.

1.4.4 Implementación

Cada componente del sistema debe ser probado durante su implementación y después de que se ha implementado, este debe ser corrido contra casos de prueba. Estas pruebas las realiza el programador, después lo hará el equipo de SQA de forma metódica, a lo que se llama pruebas unitarias.

Además de correr casos de prueba una revisión de código es una buena práctica y útil para detectar fallas en la programación. Como es obvio en esta práctica el programador guiara al equipo de SQA y una vez que se haya probado y corregido un componente se deben ir integrando con los demás que también hayan sido probados y corregidos para ir de a poco haciendo pruebas de integración y ver si las pruebas parciales exitosas tienen el mismo resultado una vez integrado el sistema y hecha la prueba total.

1.4.5 Pruebas

Existen dos tipos de pruebas: las basadas en ejecución y las basadas en no ejecución

1.4.5.1 Basadas en la no ejecución

Las pruebas de software sin correr casos de prueba se denominan pruebas basadas en la no ejecución, por ejemplo, hacer una revisión del software (leerlo) y el análisis matemático del software.

No es aconsejable que la persona que va a redactar un documento sea la única responsable en revisarlo (evitar ser juez y parte del proceso), por lo que se sugiere que una o más personas extras a la que realizo el documento hagan revisiones del mismo este es el fundamento para las técnicas de revisión como los recorridos o inspecciones. En los dos tipos de revisión un equipo de profesionales de software con el mismo nivel de conocimiento pero distintas habilidades revisan el documento que podría ser uno de especificaciones o diseño; la principal fortaleza de una revisión hecha por un equipo de expertos que las diferentes habilidades de los participantes aumentan las oportunidades de encontrar fallas y corregirlas, de esta manera aseguramos aún más la calidad de nuestros productos, inclusive los documentales de nuestro proyecto de software.

Los recorridos e inspecciones son dos tipos diferentes de revisiones, esta diferencia principalmente radica en que los recorridos tienen menos pasos y son menos formales.

Recorridos: Un equipo de recorrido constara de cuatro a seis miembros recomendablemente distribuidos así:

- ✓ Al menos un representante del equipo responsable de redactar las especificaciones.
- ✓ El responsable del flujo de trabajo de análisis.
- ✓ Un representante del cliente.
- ✓ Un representante del equipo de desarrollo.
- ✓ Y un representante del grupo de SQA.

Todos ellos deben preferentemente ser expertos en cada uno de los roles en su grupo de trabajo, de esta manera aseguramos que se puedan encontrar fallas importantes. Es decir podrán detectar aquellas fallas que tendrán un mayor impacto negativo en el proyecto.

El material que se usará en el recorrido debe ser repartido entre los participantes con anterioridad. El representante del equipo de SQA deberá dirigir el recorrido porque tiene más que perder si la revisión se realiza mal o si se les escapan fallas.

Hay que tener en cuenta que no es tarea del equipo corregir las fallas sino tan solo documentarlas para su posterior corrección. Existen dos formas de llevar a cabo un recorrido. La primera es orientada a los participantes. Estos presentan sus listas de los elementos no claros y los que piensan que son incorrectos. El representante del equipo de análisis debe responder a cada pregunta, aclarando todo lo que sea necesario para el revisor y de ser

necesario aceptar que hay una falla o explicando porque el revisor está equivocado.

La segunda forma de llevar una revisión es orientada a los documentos, una persona debe guiar en la revisión de un documento, mientras los demás revisores hacen las observaciones previamente preparadas, y probablemente este enfoque sea el más completo.

Inspecciones: Una inspección es algo más que un recorrido y tiene cinco pasos formales:

a) Las generalidades del documento que se va a inspeccionar, las debe exponer uno de los responsables del desarrollo de este documento.
b) Durante la preparación los participantes tratan de comprender con detalle el documento que se va a analizar.
c) Para comenzar la inspección u participante recorre el documento con el equipo de inspecciones, asegurándose que cada punto sea cubierto, entonces comienza el hallazgo de fallas, al igual que con los recorridos el objetivo es encontrar y documentar fallas, mas no corregirlas.
d) El re trabajo, el responsable del documento resuelve todas las fallas y problemas señalados en el informe escrito.

e) Seguimiento, la persona a cargo debe garantizar que cada uno de los aspectos documentados hayan sido resueltos de manera satisfactoria.

1.4.5.2 Basadas en la ejecución

Las pruebas sirven para demostrar que un sistema no tiene fallas presentes, sin embargo el que se hagan entregas de software "probado" no es del todo confiable, y esta contradicción se explica en lo que señala Dijkstra, "las pruebas del programa pueden ser un modo muy eficaz para mostrar la presencia de errores, pero es bastante inadecuada para mostrar su ausencia" (Dijkstra, 1972). Esto quiere decir que si se ejecuta un producto con datos de prueba y el resultado es equivocado, entonces el producto tiene una falla, pero, si el resultado es correcto el programa aun puede tener fallas, entonces concluimos que el producto corre satisfactoriamente con una serie particular de datos de prueba.

¿Pero entonces que aspectos se deben cubrir en este tipo de pruebas?, podría decir que hay 5 campos que cubrir en este tipo de pruebas y son:

1.4.6 Utilidad

La utilidad no es más saber si el producto final sirve o no al cliente; basado en las especificaciones entregadas por él, una vez que hayamos aprobado el

documento de requerimientos al menos habremos asegurado que fabricaremos un producto que hará lo que el usuario necesita conseguir.

1.4.7 Confiabilidad

Esta es la unidad de medida de la frecuencia y lo critico de una deficiencia de las deficiencias de un producto; teniendo en cuenta que una deficiencia es inaceptable bajo las condiciones operativas adecuadas y antes especificadas por el cliente.

1.4.8 Robustez

Entendemos por robustez a las capacidades operativas que tiene el producto, es decir que este producto no debe producir resultados inaceptables cuando la entrada satisface las especificaciones. Y este punto debe ser probado a lo largo de todo el ciclo de vida del software ya que con entradas permitidas en las especificaciones del mismo de esto aseguramos la robustez en el producto final.

1.4.9 Desempeño

El desempeño se basa principalmente en el tiempo de respuesta que tiene el producto bajo condiciones reales y permitidas. Un producto necesita condiciones

mínimas de hardware y software para su correcto funcionamiento, si estas condiciones no son las adecuadas el sistema no debe ni puede ser utilizado, por ejemplo, si un sistema necesita mínimo 256 Mb de memoria y se dispone de una maquina con 128 Mb dicha maquina no sirve para el sistema y este no se debe poner en este ambiente.

1.4.10 Corrección

Un producto es correcto si satisface sus especificaciones de salida independiente del uso de recursos, es decir si las condiciones de ambiente para el sistema y las entradas son correctas entonces el producto es correcto si produce la salida correcta.

1.5 Métricas y medición

Existen tres categorías de métricas de software, y estas son:

1. Métricas del producto

   Describe las características del producto tales como tamaño, complejidad, características de diseño, rendimiento, eficiencia, confiabilidad, portabilidad, etc.

2. Métricas de proceso

   Describen la eficacia y calidad del proceso que produce el software, como por ejemplo:

   - Esfuerzo requerido en el proceso
   - Tiempo de producción del producto

- Eficacia de eliminación de defectos durante el desarrollo
- Numero de defectos encontrados durante la etapa de prueba
- Maduración del proceso

3. Métricas del proyecto

   Describe las características y ejecución del proyecto

   - Numero de desarrolladores del software
   - Coste y calendario
   - Productividad

Existen cientos de métricas para el software, pero no todos son prácticos o implementables, de ahí que Lem O Ejiogu define un conjunto de atributos que deben ser tomados en cuenta en las métricas de software, tales como:

- Simple y computable, quiere decir que sean fáciles de aprender y su cálculo no exija demasiado esfuerzo o tiempo.
- Consistente en unidades y dimensiones, el cálculo de la métrica debe utilizar medidas que no den lugar a combinaciones extrañas.
- Lenguaje de programación independiente, las métricas deberían basarse en un modelo de análisis, en el modelo de diseño o en la estructura de la programación misma.
- Un mecanismo eficaz para la generación de alta calidad, la métrica debería conducir a un producto final de muy alta calidad.

Capítulo II

2 PROCESO DE TESTING

2.1 MODELOS DEL PROCESO DE TESTING

Las pruebas de software, son un elemento imprescindible y crítico para la validación de un producto o de un componente de producto de software. Sin embargo, se tiende a confundir prueba de software con validación, siendo la prueba un factor primordial del proceso de validación. Las pruebas son las últimas barreras para detectar errores, puesto que permiten encontrar fallos, como consecuencia de defectos ocasionados por errores, con el objetivo de determinar que el producto está libre de defectos y que su funcionamiento es el esperado. La prueba de software debe ser una actividad sistemática y planificada, siguiendo una serie de normas, modelos, estándares o guías, que aseguren mediante evidencia el correcto desarrollo del proceso de prueba.

El proceso de desarrollo de software describe la vida de un producto de software desde su concepción hasta su entrega, utilización y mantenimiento. Los modelos de mejora de los procesos indican pautas que las organizaciones deben seguir para mejorar sus procesos de desarrollo de software, y con esto mejorar la calidad del producto que se genera.

El proceso de prueba describe la forma en el que el producto de software debe ser probado desde su concepción; al igual que para el proceso de desarrollo, existen modelos de mejora del proceso de prueba.

Existen modelos que se basan en evaluar para mejorar, estos modelos definen mediciones a realizar para evaluar el proceso y producto, y de esta forma mejorar la calidad. Los modelos de procesos son importantes porque proporcionan un orden en el cual el proyecto debe realizar sus tareas, como se describieron en el capítulo anterior.

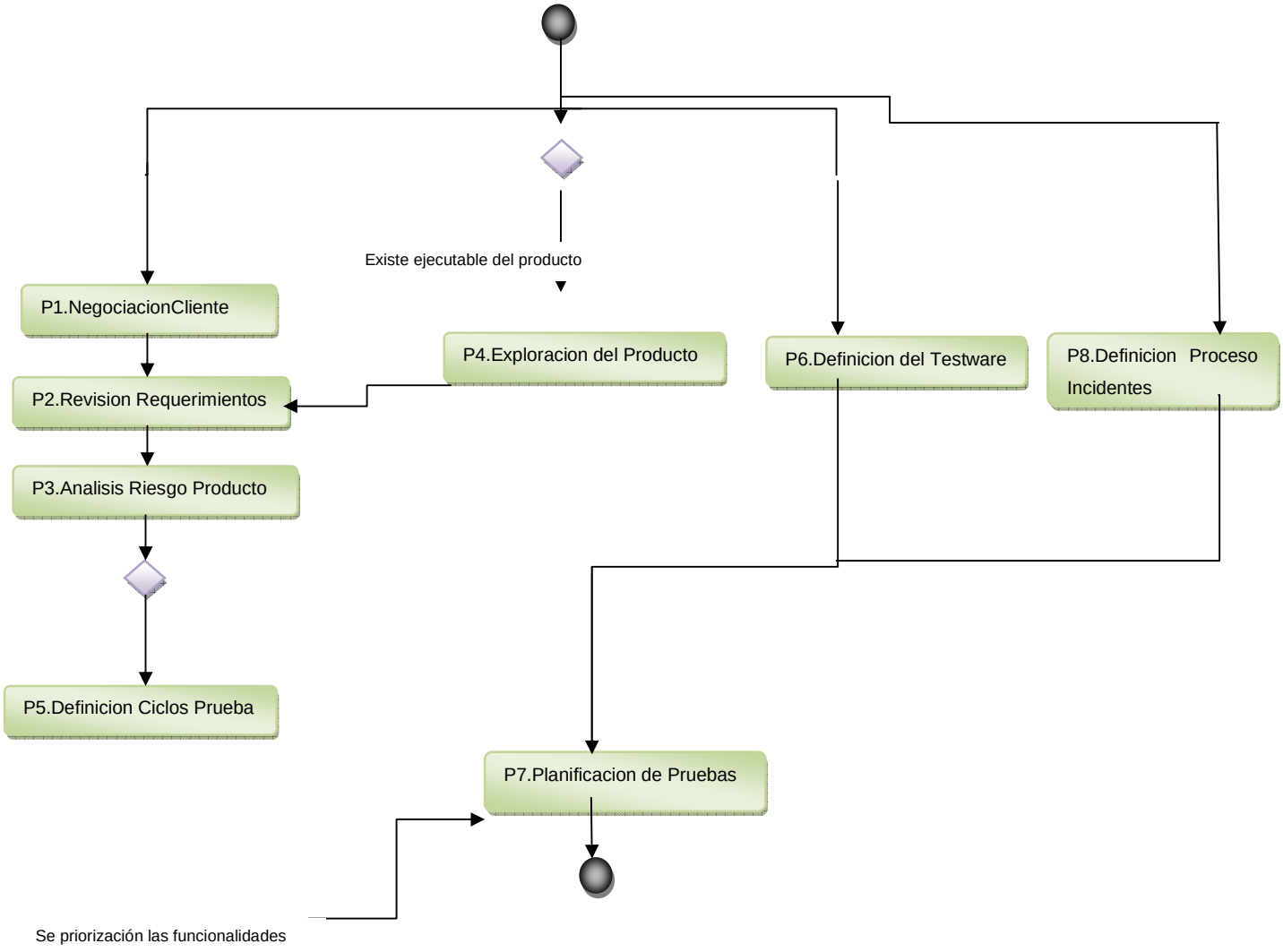
2.2 PLANEAMIENTO Y CONTROL

2.2.1 Planeamiento

El objetivo de esta etapa es planificar el proyecto de prueba, una vez que ha sido aprobado por el cliente. Se definen los ciclos de prueba y las funcionalidades a probar en cada ciclo en función del análisis de riesgo del producto. Se genera el Plan de Pruebas que resume toda la información del proyecto de prueba y las decisiones tomadas durante la etapa de Planificación.

En la Figura 5 se muestra el diagrama de las actividades involucradas en la etapa de Planificación.

Figura 5: ACTIVIDADES DE LA PLANIFICACIÓN[6]



[6]http://jummp.wordpress.com/2011/09/06/desarrollo-de-software-testing-modelo-en-v/

P1- Negociación con el Cliente: tiene como objetivo definir todas las funcionalidades aprobar y la prioridad con que será probada cada una, junto con el cliente.

P2 –Revisión de Requerimientos: se estudia las funcionalidades del producto

P3 – Análisis de Riesgo del Producto: se realiza para conocer la importancia de incluir o no una funcionalidad en el alcance de las pruebas y la prioridad con que debe ser probadas en caso de ser incluida.

P4 – Exploración del Producto: se explora el producto con el fin de entender rápidamente los requerimientos y ayudar en la actividad

P5 – Definición de los Ciclos de Prueba: se agenda los ciclos de prueba según las versiones que serán generadas por el equipo de desarrollo y se definen las funcionalidades que serán probadas en cada ciclo de prueba

P6 – Definición del Testware: se define cuáles serán los elementos que lo conformarán para el proyecto y cómo se organizarán

P7 – Planificación de las Pruebas: plan que resume toda la información del proyecto de prueba y las decisiones tomadas durante la etapa de Planificación.

P8 – Definición del Proceso de Incidentes: donde se define junto a el cliente y los desarrolladores la herramienta a seguir para el reporte de incidentes y las etapas involucradas en dicho proceso

De cada una de las actividades realizadas en la etapa de planificación de las pruebas se obtienen los siguientes artefactos generados que se detallan en la tabla 1.

Tabla 1: ACTIVIDADES Y ARTEFACTOS GENERADOS

ACTIVIDADES	ARTEFACTOS GENERADOS
P1- Negociación con el Cliente	RR – Resumen de Reunión
P2 –Revisión de Requerimientos	IP – Inventario de Pruebas
P3 – Análisis de Riesgo del Producto	IP – Inventario de Pruebas
P4 – Exploración del Producto	IP – Inventario de Pruebas
P5 – Definición de los Ciclos de Prueba	ACP – Agenda de Ciclos de Prueba
P6 – Definición del Testware	TW – Testware
P7 – Planificación de las Pruebas	PP – Plan de Pruebas
P8 – Definición del Proceso de Incidentes	PI – Proceso de Incidentes

Elaborado por: TULIO NAVARRETE

2.2.2 Controles

Para mejora de la calidad y la mejora de las capacidades de los procesos, se han creado Modelos de Calidad que ayudan a tener estándares y controles al momento de mantener el mejoramiento continuo para un producto de software, sobre todo en la etapa de testing.

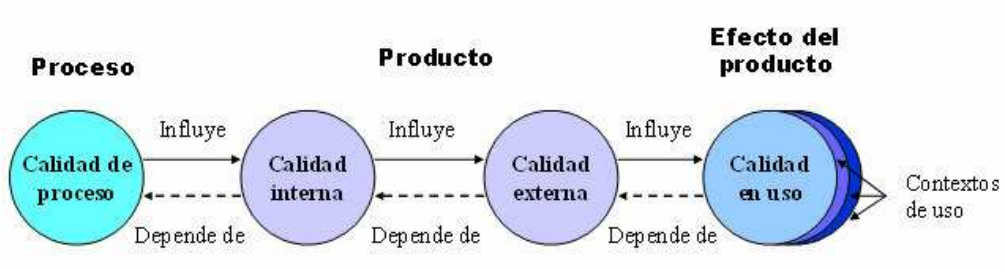
2.2.3 ISO/IEC 9126

Describe un modelo de calidad para un producto de software en dos partes: a) calidad interna y calidad externa, y b) calidad en el uso. La primera parte del modelo especifica seis características para la calidad interna y externa. La segunda parte del modelo especifica cuatro características de calidad en el uso. La calidad en el uso es

el efecto combinado para el usuario de las seis características de la calidad del producto de software.

La calidad de proceso contribuye a mejorar la calidad del producto, y la calidad del producto contribuye a mejorar la calidad en el uso. Por lo tanto, evaluar y mejorar el proceso es una forma de mejorar la calidad del producto, y evaluar y mejorar la calidad del producto es una forma de mejorar calidad en uso. De la misma manera, evaluar la calidad en el uso ayuda a mejorar el producto, y la evaluación del producto ayuda a mejorar el proceso.

Figura 6: CALIDAD EN EL CICLO DE VIDA[7]



2.3 ANÁLISIS COMPARATIVO DE LOS PRINCIPALES MÉTODOS DE TESTING DE SOFTWARE

Hoy en día, debido al aumento del tamaño y la complejidad del software, el proceso de testing se ha convertido en una tarea vital dentro del proceso de desarrollo de cualquier sistema software. Para analizar cómo se pueden aplicar métodos de

[7]http://www.ces.com.uy/documentos/imasd/

testing, es necesario realizar una clasificación de los tipos de pruebas existentes que permitan analizar la situación actual.

2.3.1 Prueba Estructural

Es también llamada de caja blanca o basada en el código, esta estrategia implica que se tiene en cuenta la estructura del código fuente del programa para seleccionar casos de prueba.

En muchas ocasiones se pone tanto énfasis en la estructura del código que se ignora la especificación del programa, convirtiendo al testing en una tarea un tanto desprolija e inconsistente.

> *"Se dice que el testing estructural prueba lo que el programa hace y no lo que se supone que debe hacer"*[8].

2.3.2 *Prueba Funcional*:

El objetivo es validar cuando el comportamiento observado del software probado cumple o no con sus especificaciones; la prueba funcional toma el punto de vista del usuario. Las funciones son probadas ingresando las entradas y examinando las salidas, la estructura interna del programa raramente es considerada.

[8] http://www.sistedes.es/TJISBD/Vol-3/No-4/articles/pris-09-calvo-cmmidev.pdf

> "*Se dice que el testing basado en modelos prueba lo que el programa se supone que debe hacer, y no lo que el programa hace"*[9]

Es muy importante remarcar que estas estrategias no son opuestas sino complementarias. El testing debería estar guiado fundamentalmente por técnicas de MBT[10] pero complementadas con herramientas de análisis de cubrimiento de sentencias de forma tal que los casos generados mediante MBT cubran al menos todas las líneas de código.

2.3.3 Prueba de integración:

Es el proceso en el cual los componentes son agregados para crear componentes más grandes. Es el testing realizado para mostrar que aunque los componentes hayan pasado satisfactoriamente las pruebas unitarias.

2.3.4 Prueba de Aceptación

Es el proceso de comparar el programa contra sus requerimientos iníciales y las necesidades reales de los usuarios. Es realizado generalmente por el cliente o el usuario final.

2.3.5 Prueba del sistema

Refiere al comportamiento del sistema entero. La mayoría de las faltas funcionales deben haber sido identificadas ya durante las pruebas de unidad e integración. La prueba del sistema generalmente se considera apropiada para probar los

[9] http://www.sistedes.es/TJISBD/Vol-3/No-4/articles/pris-09-calvo-cmmidev.pdf

[10] MBT: es la automatización del testing de caja negra.

requerimientos no funcionales del sistema, tales como seguridad, desempeño, exactitud, y confiabilidad.

2.3.6 Pruebas de Regresión:

Su objetivo es verificar que no ocurrió una regresión en la calidad del producto luego de un cambio, asegurándose que los cambios no introducen un comportamiento no deseado u errores adicionales, esto implican la re ejecución de alguna o todas las pruebas realizadas anteriormente.

2.3.7 Prueba unitaria:

Es el proceso de probar los componentes individuales subprogramas o procedimientos de un programa. El propósito es descubrir discrepancias entre la especificación de la interface de los módulos y su comportamiento real.

Hay tres tipos de pruebas de regresión:

- ✓ Regresión de defectos solucionados: Cuando se reporta un defecto y vuelve una nueva versión que supuestamente lo soluciona, el objetivo es probar que no fue solucionado.
- ✓ Regresión de defectos viejos: Se prueba que un cambio en el software causó que un defecto que ya había sido solucionado vuelva a aparecer.
- ✓ Regresión de efectos secundarios: Implica volver a probar una parte del producto. El objetivo es probar que el cambio ha causado que algo que funcionaba ya no funciona.

2.3.8 Prueba de Humo:

Son un conjunto de pruebas aplicadas a cada nueva versión, su objetivo es validar que las funcionalidades básicas de la versión se cumplen según lo especificado. Estas pruebas buscan grandes inestabilidades o elementos clave faltantes o defectuosos, que hacen imposible realizar la prueba como fue planificado para la versión.

A continuación se presenta un cuadro comparativo basado en el aprendizaje, implantación, funcionalidad de las pruebas, en el cual se califica de acuerdo a la tabla que hace referencia.

Tabla 2: NIVEL DE PUNTUACIÓN[11]

PUNTAJE	DESCRIPCION
1	Excelente
2	Muy Bueno
3	Regular
4	Malo
5	Deplorable

Elaborado por: Tulio Navarrete

[11]Referencia tomada de Ingeniería de Software; IanSommerville – 7 Edición ; pag 364

Tabla 3: NIVEL DE PUNTUACION[12]

	Aprendizaje	Implantación	Funcionalidad	TOTAL
Prueba Estructural	2	2	3	7
Prueba Funcional	2	2	2	6
Prueba unitaria	1	2	2	5
Prueba de integración	2	2	2	6
Prueba de Aceptación	1	1	1	3
Prueba del sistema	2	2	1	5
Pruebas de Regresión	3	3	2	8
Prueba de Humo	3	2	1	6

Elaborado por: Tulio Navarrete

Se ha seleccionado al aprendizaje, implantación y funcionalidad como ejes fundamentales del proceso de testing tomando en cuenta todos los actores que deben estar involucrados en este proceso (desarrolladores y usuario final) y tomando en cuenta también lo que el proceso deja como experiencia a sus actores para el futuro, no se quiere con esto decir que un proceso es mejor que otro, solo demostrar que mientras más involucrados estén los todos los actores en el proceso, serán mejores los resultados y el margen de error en el producto final será mínimo con tendencia a nulo.

[12]Referencia tomada de Ingeniería de Software; IanSommerville – 7 Edición ; pag 364

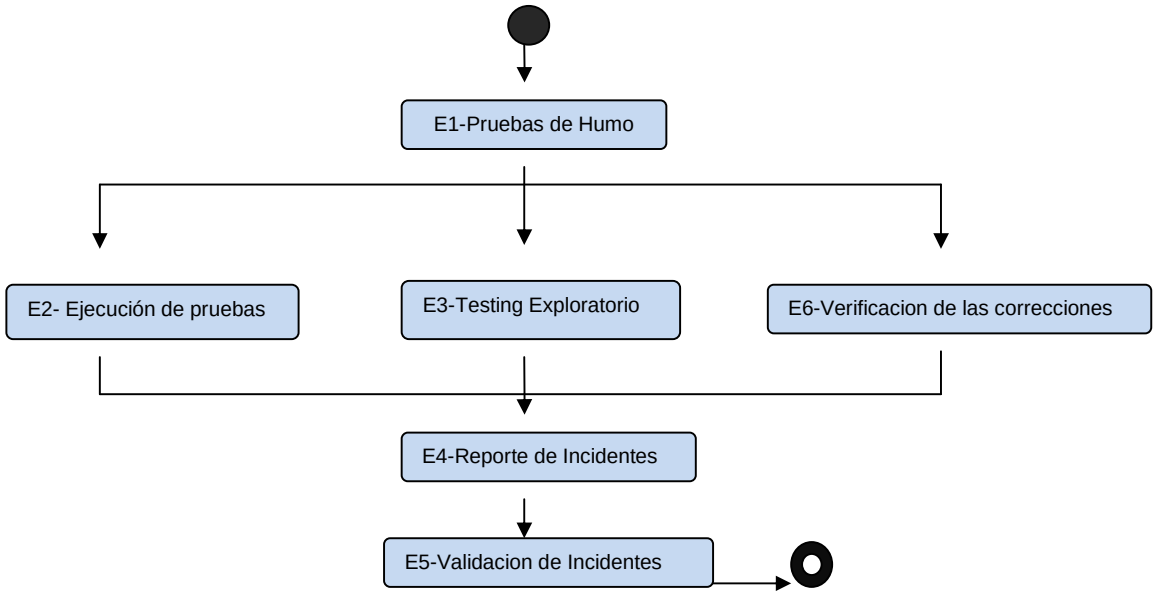
De acuerdo al análisis comparativo las pruebas de aceptación son los que mejores resultados se tienen porque se prueba la funcionalidad del sistema de software, con las necesidades del cliente en una prueba conjunta entre tester y cliente.

2.4 IMPLEMENTACIÓN Y EJECUCIÓN

El objetivo es contrastar el comportamiento esperado del software con su comportamiento real, analizar las diferencias y reportar los resultados.

En la Figura 7 se muestra el diagrama con las actividades de la Ejecución de las Pruebas.

Figura 7: ACTIVIDADES DE LA EJECUCIÓN DE LAS PRUEBAS[13]



[13] http://www.ces.com.uy/documentos/imasd/

E1- Pruebas de Humo: se valida que las funcionalidades básicas de la versión a probar están presentes.

E3- Testing Exploratorio: se ejecutan las pruebas del producto. Para los incidentes encontrados en ciclos anteriores, que fueron corregidos para la versión que se está probando

E6- Verificación de las Correcciones: incidentes encontrados

E5 – Validación de los Incidentes: incidentes reportados son validados por el cliente y los desarrolladores esta actividad.

De cada una de las actividades realizadas en la etapa de ejecución de las pruebas se obtienen los siguientes artefactos generados que se detallan en la tabla 4.

Tabla 4: ACTIVIDADES Y ARTEFACTOS GENERADOS EN LA ETAPA DE EJECUCION

ACTIVIDADES	ARTEFACTOS GENERADOS
E1 – Pruebas de Humo	RP – Reporte de Ejecución de las Pruebas
E2 – Ejecución de las Pruebas	RP – Reporte de Ejecución de las Pruebas
E3 – Testing Exploratorio	RP – Reporte de Ejecución de las Pruebas
E4 – Reporte de Incidentes	RI – Reporte de Incidente
E5 – Validación de los Incidentes	RI – Reporte de Incidente
E6 – Verificación de las Correcciones	RP – Reporte de Ejecución de las Pruebas

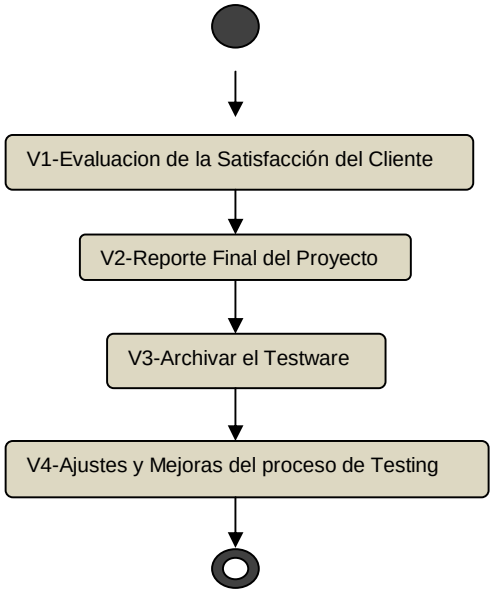
Elaborado por: Tulio Navarrete

2.5 EVALUACIÓN Y REPORTE

Esta etapa tiene como objetivo conocer el grado de satisfacción del cliente, realizar el informe final, evaluar el proceso de prueba para su mejora y almacenar los elementos del proyecto de prueba para su uso en proyectos posteriores.

En la Figura 8 se muestra el diagrama de actividades para la etapa Evaluación.

Figura 8: ACTIVIDADES DE LA ETAPA DE EVALUACIÓN[14]



[14] http://www.ces.com.uy/documentos/imasd/

V1- Evaluación de la Satisfacción del Cliente tiene como objetivo conocer el grado de satisfacción del cliente con el proyecto de prueba

V2 – Mejoras y Ajustes al Proceso de Prueba: el equipo de prueba mejora el proceso para próximos proyectos

V3- Reporte Final del Proyecto: se realiza el informe final del proyecto de prueba

V4- Archivar el Testware: se almacenan los elementos del Testware para ser consultados en próximos proyectos.

2.6 CLAUSURA DE LAS ACTIVIDADES

Las lecciones deberían ser aprendidas de proyectos anteriores, es decir; la raíz de la causa de defectos encontrados en otros proyectos, los procesos pueden ser mejorados, lo que debe evitar que esos defectos vuelvan a producirse y así, mejorar la calidad de sistemas futuros.

Las actividades de cierre del test recogen todos los datos de las actividades de test completadas para consolidar experiencia, testware, hechos y números. Cuando un sistema software se publica, o se ha completado un proyecto de test, o se ha completado una etapa de mantenimiento.

Las actividades de cierre de test son:

- ✓ Chequear que entregables planificados han sido entregados, el cierre de los informes de incidencias o aumento de los registros de cambio para cualquier informe que continúe abierto y documentación de la aceptación del sistema.
- ✓ Finalización y archivo del testware, el entorno de test y su infraestructura para su futura reutilización.
- ✓ Entrega del testware a la organización de mantenimiento.
- ✓ Análisis de las lecciones aprendidas para futuras revisiones y proyectos, y la mejora de la experiencia en test.

Capítulo III

3 DOCUMENTACIÓN DEL PROCESO TESTING

En un proyecto de desarrollo de software existe un conjunto de documentos o instrumentos asociados a cada una de las fases del ciclo de vida: planificación, análisis, diseño, construcción, se puede considerar el proceso de testing como un proyecto que se ejecuta en paralelo con el desarrollo y en el que se pueden distinguir tres grandes etapas:

- ✓ Preparación de las pruebas.
- ✓ Ejecución de las pruebas.
- ✓ Finalización de las pruebas.

En cada una de estas fases hay que generar la documentación apropiada, lo cual puede ser complicado si no se tiene una referencia adecuada.

3.1 ESPECIFICACIÓN DE REQUERIMIENTOS

La ingeniería de requisitos facilita el mecanismo apropiado para comprender lo que quiere el cliente, analizando necesidades, confirmando su viabilidad, negociando una solución razonable, especificando la solución sin ambigüedad, validando la

especificación y gestionando los requisitos para que se transformen en un sistema operacional.

Los requerimientos para un sistema de software determinan lo que hará el sistema y definen las restricciones de su operación e implementación.

En algunos casos, un requerimiento se visualiza como una declaración abstracta de alto nivel de un servicio que debe proveer el sistema o como una restricción de éste.

3.1.1 Requerimientos funcionales

Los requerimientos funcionales de un sistema describen la funcionalidad o los servicios que se espera que éste provea. Estos dependen del tipo de software y del sistema que se desarrolle y de los posibles usuarios del software. Cuando se expresan como requerimientos del usuario, habitualmente se describen de forma general mientras que los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etc.

3.1.2 Requerimientos no funcionales

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema.

Una gran variedad de organizaciones han definido estándares para los documentos de requerimientos.

El IEEE sugiere la siguiente estructura para los documentos de requerimientos.

1. Introducción

- ✓ Propósito del documento de requerimientos
- ✓ Alcance del producto
- ✓ Definiciones, acrónimos y abreviaturas
- ✓ Referencias
- ✓ Resumen del resto del documento.

2. Descripción general

- ✓ Perspectiva del producto
- ✓ Funciones del producto
- ✓ características del usuario
- ✓ Restricciones generales
- ✓ Suposiciones y dependencias

3. Requerimientos específicos: Cubren los requerimientos funcionales, no funcionales y de interfaz. Obviamente, ésta es la parte más sustancial del documento, pero debido a la amplia variabilidad en la práctica organizacional, no es apropiado definir una estructura estándar para esta sección.

4. Apéndices: Es la sección al final que incluye la información complementaria que puede ser de interés o utilidad, se puede utilizar también para apoyar la cualificación de los requerimientos, y para aumentar la credibilidad

5. Índice: Muestra la manera en que el texto se distribuye dentro de los márgenes del trabajo.

Los requerimientos pueden documentar las interfaces externas, describir la funcionalidad y el desempeño del sistema, especificar los requerimientos lógicos de la base de datos, las restricciones de diseño, las propiedades emergentes del sistema y las características de calidad.

Aunque el estándar IEEE no es ideal, contiene una buena ayuda de cómo tratar los requerimientos y evitar problemas. Es muy general para que pueda ser estándar de una organización. Sin embargo, se puede transformar y adaptar para definir un estándar que se ajuste a las necesidades de una organización en particular.

La información que se incluya en un documento de requerimientos debe depender del tipo de software a desarrollar y del enfoque de desarrollo que se utilice.

3.2 DOCUMENTO DE LA ESTRATEGIA DE TESTING

La estrategia de pruebas busca que el producto de software que se está construyendo o modificando, reúna los requerimientos de lógica del negocio que el cliente ha podido realizar mediante el debido contrato de desarrollo de software. Son las líneas guías del equipo de pruebas de un proyecto de desarrollo de software, a

menudo estas son escritas por el director del equipo de pruebas , en un documento que formalmente será entregado al director del proyecto , para su posterior revisión y aprobación.

Se debe tener claro en que dicho documento no debe ser visto como un entregable más, este debe ser visto como un artefacto especialmente creado para reunir las ideas más representativas del proceso de pruebas que se llevara a cabo.

Se añade un ejemplo de un documento de estrategia de pruebas en la parte de Anexos.

3.3 ESPECIFICACIÓN DE INTERFACES

En este tipo de implementaciones, parte de un análisis y levantamiento de información de los procesos involucrados que luego es plasmado en un flujo con distintas ramas de acuerdo a la complejidad del proceso, este flujo contiene a su vez formularios y reglas que indican porque rama del flujo debe seguirse de acuerdo a las condiciones de entrada.

- ✓ Considerar probar las reglas de negocio que indiquen algún comportamiento distinto en una rama.
- ✓ Considerar las interfaces que hubiera con entes externos al flujo, las tramas que sirven de nexo.

- ✓ Las notificaciones deben probarse considerando casos aparte para que no sean cuellos de botella en los caos sobre las ramas del flujo y además para tener mapeadas las notificaciones y ser más efectivo en las pruebas.
- ✓ Los temas de Perfiles y accesos como siempre se han probado.
- ✓ Si existieran Servicios externos que envían información al flujo y viceversa es considerar validar en un ambiente apropiado dado que la lógica interna de los servicios puede ser más compleja que lo requerido por el flujo, esto en el caso que estos servicios se hayan construido o modificado para ser consumidos por el flujo.

3.4 TEST PLAN

El plan de pruebas constituye el plan maestro que va a dirigir los esfuerzos de testing de un proyecto determinado. Se debe contemplar los siguientes aspectos:

- ✓ Qué elementos y funcionalidades del producto software van a ser probados y cuáles no, es decir el alcance de las pruebas.
- ✓ Quién va a realizar las pruebas, asignar responsabilidades y qué recursos se necesitan en cuanto a personas, software, hardware y formación.

Los aspectos relativos a la calidad de las pruebas; los tipos de pruebas a realizar de componentes, de integración, y las técnicas elegidas para diseñar las pruebas, así como el nivel de cobertura y los criterios de salida; es decir, los principales riesgos a

tener en cuenta, en especial las circunstancias bajo las cuales se parará o se reiniciará el proceso de testing.

La especificación del diseño de pruebas es el primer paso en el desarrollo real de las pruebas. Este documento especifica las características y funcionalidades que se quieren probar del producto software condiciones de test o requisitos de test y su priorización, así como los criterios de éxito/fallo de las pruebas.

El documento de especificación de diseño de pruebas ayuda a determinar en una fase temprana dónde se quieren centrar los esfuerzos de testing, de tal forma que después no se malgasten energías en crear casos de prueba para elementos que no merecen la pena.

3.5 DOCUMENTO DE MONITOREO DE TESTING

Los procesos son llevados a cabo por personas, por lo que sus descripciones deben estar disponibles a todos los miembros de la organización. Se debe contar con documentación accesible del proceso, en un formato imprimible.

Los cambios introducidos en el proceso requieren generar nuevamente la documentación, si el proceso tiene gran número de actividades, cada cambio puede resultar difícil de realizar, pudiendo quedar inconsistente la información en distintos elementos del proceso.

Se añade un ejemplo de un documento de monitoreo de pruebas en la parte de Anexos.

3.6 ESPECIFICACIÓN DE CASOS DE PRUEBA

Un caso de prueba corresponde a un conjunto de datos de entrada con el que se ejecutará la prueba y un conjunto de criterios de aceptación para el caso de prueba.
Las condiciones o requisitos de test suelen estar especificadas de forma muy vaga. Un caso de test realiza una especificación más detallada indicando:

- ✓ Los datos de entrada concretos que hay que utilizar.
- ✓ Cuál es el resultado esperado tras la ejecución de la prueba.

Otros elementos relevantes que deben indicar el documento de casos de prueba son:

- ✓ Las precondiciones que se han de dar antes de la ejecución del caso de prueba.
- ✓ Interdependencias entre los casos de test.

3.7 ESPECIFICACIÓN DE PROCESOS DE PRUEBA

El proceso de pruebas propuesto está basado un modelo en V, es decir que en forma paralela desarrollo del producto se van creando los planes de prueba, pruebas y

casos de prueba con sus criterios de aceptación para que al término del desarrollo se tengan listas las pruebas que deben llevarse a cabo. Estas pruebas pueden ser unitarias, ejecutadas por el desarrollador o de aceptación las que usualmente involucran pruebas a todo el sistema realizadas por usuarios finales.

Este documento especifica aspectos como:

- ✓ Los pasos detallados de cómo ejecutar cada test y el orden de ejecución.
- ✓ La configuración exacta del entorno de pruebas.

3.8 REPORTE DIARIO DE PRUEBAS

Un objetivo fundamental del proceso testing es proporcionar información acerca del sistema que se está probando.

El registro de pruebas documenta los aspectos relativos a la ejecución de las pruebas: qué test se han ejecutado, quién y cuándo los ha ejecutado, en qué orden, en qué entorno, si el test ha pasado o ha fallado. En este documento es importante también, asociar la información de ejecución de cada test con versiones específicas del software en prueba para garantizar la trazabilidad.

La información recogida en el registro de pruebas permite conocer el progreso de la fase de ejecución de pruebas y tomar decisiones acerca de si el software está listo para pasar a la siguiente fase.

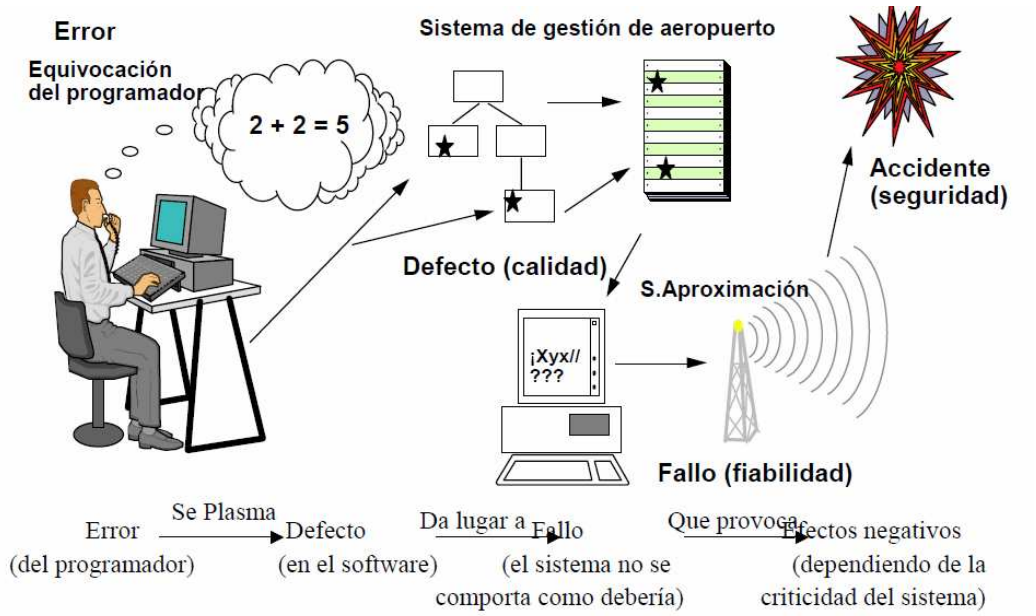
Se añade un ejemplo de un documento reporte diario de pruebas en la parte de Anexos.

3.9 REPORTE DE ERRORES O BITÁCORA DE INCIDENCIAS

Hay que resaltar la referencia al término “incidente”; un incidente no tiene porqué deberse necesariamente a un defecto en el sistema. Un incidente representa una discrepancia entre los resultados esperados y los resultados obtenidos. Esta discrepancia puede deberse a varios motivos, como un defecto, un error en la especificación de los casos de prueba, una mala interpretación de los requisitos, etc.

Un error se puede definir como una acción humana que produce o genera un resultado incorrecto; un defecto es la manifestación de un error en el software, un defecto es encontrado porque causa una falla, la cual es una desviación del servicio o resultado esperado.

Figura 9: RELACIÓN ENTRE ERROR, DEFECTO Y FALLO[15]



El informe de incidentes debe contener toda la información necesaria para la identificación y resolución del incidente: entradas utilizadas, resultados esperados, resultados obtenidos, paso del procedimiento en el que se ha producido el incidente, configuración del entorno, valoración del impacto, etc.

Se añade un ejemplo de un documento Reporte De Errores o Bitácora De Incidencias en la parte de Anexos.

[15] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf

Capítulo IV

4 HERRAMIENTAS DE TESTING

4.1 INTRODUCCIÓN

Existe una valoración importante dentro de los probadores de software profesional relacionada con el conocimiento teórico y práctico. Es importante esclarecer, métodos, conceptos y estructuras de normas internacionales para desarrollar una eficiente tarea de QA; un conocimiento más que necesario en el competitivo mercado actual y en el cumplimiento de los requerimientos de calidad del contexto internacional, el manejo de herramientas y posibilidades de implementación; es un conocimiento de un gran valor agregado.

Algunas herramientas evidentemente soportan una actividad, otras pueden soportar más de una actividad, pero son clasificadas según la actividad con la cual ellas son más estrechamente asociadas. Algunas herramientas comerciales a menudo dan soporte para solo un tipo de actividad: otros vendedores de herramientas comerciales ofrecen un conjunto o familias de herramientas que proporcionan soporte para muchas o todas esas actividades.

Las herramientas de prueba pueden mejorar la eficiencia de las actividades de prueba mediante la automatización de tareas repetitivas. Las herramientas de prueba

pueden también mejorar la fiabilidad de las pruebas mediante, la automatización de la comparación de numerosos datos o la simulación de comportamientos.

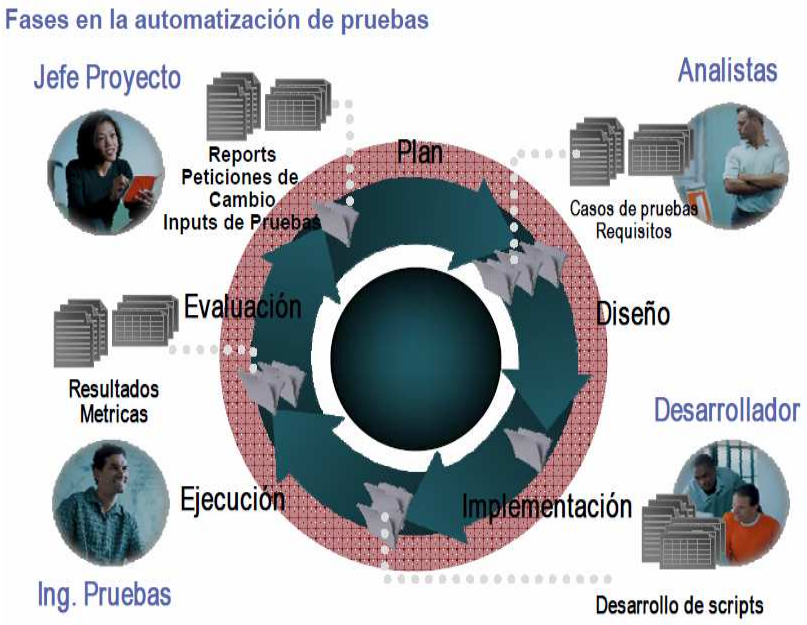
4.2 HERRAMIENTAS AUTOMÁTICAS DE TESTING

Automatización de pruebas es una de las inversiones que ha producido los mejores y más satisfactorios resultados

- ✓ Inversión a medio-largo plazo
- ✓ Única actividad que nos proporciona una estimación real de la calidad de nuestra aplicación

Desarrollo iterativo, probar en cada iteración, cada vez que se realiza un cambio.

Figura 10: FASES DE AUTOMATIZACIÓN[16]



[16] http://www.ati.es/IMG/pdf/IBM.pdf

4.3 HERRAMIENTAS DE ANÁLISIS DINÁMICO

Las herramientas de análisis dinámicas encuentran defectos que son evidentes solamente cuando el software se está ejecutando, tales como dependencias de tiempo o pérdida de memoria. Son típicamente usadas en pruebas de componentes y pruebas de integración de componentes.

4.4 HERRAMIENTAS DE MEDICIÓN

Las herramientas de medición de cobertura pueden ser intrusiva o no intrusiva dependiendo de las técnicas de medida usadas, qué es lo que se mide y el lenguaje del código. Las herramientas de cobertura de código mide el porcentaje de los tipos específicos de estructura de códigos que se han realizado.

Estas herramientas muestran perfectamente cómo el tipo de estructura medido ha sido realizado por un conjunto de pruebas.

4.5 HERRAMIENTAS DE RENDIMIENTO DE TESTING

Dentro de su política de calidad establecida con carácter general y, específicamente, como complemento al plan de calidad de desarrollo de software, ofrece un sistema de prueba de rendimiento de las aplicaciones entregadas con el fin de garantizar el cumplimiento de los requisitos mínimos de servicio demandados por el cliente y, al

mismo tiempo, facilitar el paso de los productos obtenidos al entorno de producción con las máximas garantías posibles de éxito.

El resultado obtenido consiste en una serie de informes que reflejan el rendimiento del sistema en distintos escenarios.

4.6 HERRAMIENTAS DE COMPRENSIÓN Y ANÁLISIS DE CÓDIGO FUENTE

Se aplica directamente sobre el código fuente tal cual, sin transformaciones previas ni cambios de ningún tipo. La idea es que, en base a ese código fuente, podamos obtener información que nos permita mejorar la base de código manteniendo la semántica original.

El analizador de código, recibirá el código fuente de nuestro programa, lo procesará intentando averiguar qué es lo que queremos que haga y nos dará sugerencias con las que poder mejorar ese código.

Figura 11: ANALISIS DE CODIGO FUENTE[17]



[17] http://raulexposito.com/files/documentos/AnalisisEstaticoCodigo.pdf

Estas herramientas incluyen, por un lado, analizadores léxicos y sintácticos que procesan el código fuente y, por otro, un conjunto de reglas que al aplicar sobre determinadas estructuras nuestro código fuente posee una estructura concreta que el analizador considere como "*mejorable*" en base a sus reglas nos lo indicará y nos sugerirá una mejora. Básicamente ganamos en facilidad de mantenimiento y de desarrollo ya que su objetivo es minimizar la deuda técnica de nuestros proyectos, y es que algunas de las funciones de los analizadores consisten en encontrar partes del código que puedan:

- ✓ Reducir el rendimiento,
- ✓ Provocar errores en el software,
- ✓ Complicar el flujo de datos,
- ✓ Tener una excesiva complejidad,
- ✓ Suponer un problema en la seguridad.

4.7 HERRAMIENTAS DE ANÁLISIS ESTÁTICO

Las herramientas de análisis estático ayudan a los desarrolladores, probadores y personal encargado de la garantía de calidad, a encontrar defectos antes de la comprobación dinámica. Los importantes propósitos incluyen:

- ✓ La aplicación de códigos estándares.
- ✓ El análisis de estructuras y dependencias
- ✓ Ayuda en la comprensión de los códigos.

Las herramientas de análisis estático pueden calcular medidas desde el código, el cual puede dar valiosa información, por ejemplo, para planificación o análisis de riesgos.

Las herramientas de análisis estático aplicado a un código fuente pueden imponer códigos estándares, pero si se aplican a códigos existentes pueden generar muchos mensajes. Los mensajes de aviso no detienen el código que está siendo traducido a un programa ejecutable, pero perfectamente debería ser dirigido para que el mantenimiento del código sea más fácil en el futuro. Una implementación gradual con filtros iníciales para excluir algunos mensajes sería una efectiva aproximación.

Tabla 5: Herramientas de análisis estadístico y principales características

Herramientas de carga y rendimiento	Web Functional/Regression Test Tools	Validadores de HTML	Comprobadores de Links	Herramientas de comprobación seguridad
JMeter(es una herramienta Open Source realizada en java, permite realizar simulaciones de gran carga en el servidor, red o aplicación para comprobar su fuerza y para analizar el rendimiento ante diferentes tipos de sobrecarga)	JUnit(permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera)	W3C Validator(Es muy exigente y en ocasiones es difícil cumplirlo al 100%)	W3C Link Checker(permite analizar una web en busca de enlaces rotos.)	Powerfuzzer(permite a los administradores crear tests personalizados para sus aplicaciones Web con la intención de detectar agujeros de seguridad)

JCrawler (Admite redirecciones HTTP y cookies. Es independiente de la plataforma, posee un modo consola y es sencillo de configurar)	HttpUnit(librería java Open Source para realizar testeos de caja negra en aplicaciones web HTTP y servlets)	HTML Validator(Plugin para firefox mediante el que podremos validar el HTML de la web.)	LinkChecker(Plugin para firefox mediante el que se valida si los links de una web funcionan correctamente)	Netsparker(producto de análisis de vulnerabilidades web, inclusión local y remota de ficheros, inyección de comandos)
SOLEX(permite grabar la sesión de un usuario, permitiéndole configurar en función a diferentes parámetros)	JWebUnit(es una extensión de los frameworksjunit y htmlunit, especialmente indicado para probar la interfaz de la aplicación. Con este framework, las pruebas podrán interactuar con la aplicación web, simulando las diferentes acciones del usuario y revisando la interfaz que genera la aplicación en cada momento)			Zero Day Scan(es un servicio online gratuito que detecta las vulnerabilidades de una Web, genera un resumen de resultados. Requiere la confirmación del propietario de la página.)
	Jameleon(Divide la aplicación en características, crea casos de prueba para cada una de ellas. Permite reducir enormemente el costo de crear Testcases)			Nessus(programa de escaneo de vulnerabilidades en diversos sistemas operativos, comienza escaneando los puertos o con su propio escaneador de puertos para buscar puertos abiertos y después intentar varios exploits

				para atacarlo)

Elaborado por: TULIO NAVARRETE

Capítulo V

5 ANÁLISIS DE REQUERIMIENTOS PARA EL TESTING DE SOFTWARE

5.1 INTRODUCCIÓN

La mayoría de proyectos de software son complejos, y la estrategia primaria para superar la complejidad, es la descomposición. La estrategia es dividir el problema en unidades más pequeñas que sean manejables. Un enfoque tradicional para realizar esto fue el análisis y diseño estructurados, donde se trata de descomponer el problema en funciones o procesos.

Los requerimientos son una descripción de las necesidades o deseos de un producto, una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

Para asegurar la calidad de los desarrollos de software se tiene que poner a prueba lo que se produce. El software es construido por seres humanos y los humanos cometen errores por lo tanto es importante testear un software ya que este paso evita que ocurran defectos o errores los cuales se pueden llegar a dar por diversas

razones como la infraestructura para el desarrollo del software y que la creatividad de desarrollador no coincida con las especificaciones previamente.

El software puede empezar a probarse desde su diseño o bien desde que se escribe el primer módulo, se termina de testear cuando la aplicación fue correctamente revisada con los pasos determinados es decir los casos de prueba. De esta manera cada parte del software estará construida correctamente cumpliendo con los requisitos funcionales definidos por el usuario y los requisitos no funcionales definidos por los analistas.

Es fundamental en el análisis del proceso de testing tener un control del software, generar planillas de casos de prueba que permitan no solo facilitar la labor del tester a la hora de probar los hipotéticos errores del programa sino que también suministren un rápido acceso a los pasos que se realizaron para encontrar dichos bugs. Es importante definir en qué contexto se dio este error, para poder entender todo lo que estaba pasando en el momento que se estaba haciendo la prueba.

Son necesarios revisar todos estos requerimientos para garantizar la calidad del software que se está brindando y que cumple con las especificaciones dadas por el cliente. Aumentar la satisfacción y obtener un crecimiento, prestigio y reconocimiento como organización por brindar un producto de calidad.

5.2 REQUERIMIENTOS FUNCIONALES

Son declaraciones de los servicios que proveerá el sistema, de la manera en que éste reaccionará a entradas particulares. En algunos casos, los requerimientos funcionales de los sistemas también declaran explícitamente lo que el sistema no debe hacer.

Los requerimientos funcionales de un sistema describen la funcionalidad o los servicios que se espera que éste provea. Estos dependen del tipo de software y del sistema que se desarrolle y de los posibles usuarios del software. Cuando se expresan como requerimientos del usuario, habitualmente se describen de forma general mientras que los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etc.

En principio, la especificación de requerimientos funcionales de un sistema debe estar completa y ser consistente. La compleción significa que todos los servicios solicitados por el usuario están definidos. La consistencia significa que los requerimientos no tienen definiciones contradictorias. En la práctica, para sistemas grandes y complejos, es imposible cumplir los requerimientos de consistencia y compleción. La razón de esto se debe parcialmente a la complejidad inherente del sistema y parcialmente a que los diferentes puntos de vista tienen necesidades inconsistentes; estás inconsistencias son obvias cuando los requerimientos se especifican por primera vez. Los problemas emergen después de un análisis profundo. Una vez que éstos se hayan descubierto en las diferentes revisiones o en

las fases posteriores del ciclo de vida, se deben corregir en el documento de requerimientos.

Estos requerimientos son una parte fundamental dentro de las pruebas de testing porque expresan las necesidades del cliente, lo que realmente espera obtener del sistema de software.

Para realizar las pruebas de los requerimientos funcionales se pueden optar por varias opciones, la más utilizada es por medio de un checklist de cada una de las funcionalidades que se espera alcanzar del sistema de software, estas pruebas se las realiza conjuntamente con el cliente y los usuarios involucrados para ir cubriendo todos los requerimientos antes solicitados.

5.3 REQUERIMIENTOS NO FUNCIONALES

Son aquellos requerimientos que no se refieren directamente a las funciones específicas que entrega el sistema, sino a las propiedades emergentes de éste como la fiabilidad, la respuesta en el tiempo y la capacidad de almacenamiento. De forma alternativa, definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y la representación de datos que se utiliza en la interface del sistema.

Los requerimientos no funcionales surgen de la necesidad del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas de software o hardware o a factores externos

como los reglamentos de seguridad, las políticas de privacidad, etc.

Los Requerimientos No Funcionales generales estarán enmarcados en los siguientes aspectos:

5.3.1 Escalabilidad

- ✓ El diseño debe contemplar el uso óptimo de recursos tales como conexiones a la base de datos.
- ✓ Contemplar en el diseño la clara partición entre datos, recursos y aplicaciones para optimizar la escalabilidad del sistema.
- ✓ Debe contemplar requerimientos de crecimiento para usuarios tanto internos como externos.

5.3.2 Disponibilidad

- ✓ La disponibilidad del sistema debe ser continua con un nivel de servicio para los usuarios de 7 días X 24 horas, garantizando un esquema adecuado que permita ante una posible falla de la solución en cualquiera de sus componentes, contar con una contingencia, generación de alarmas.
- ✓ Debe contemplar requerimientos de confiabilidad y consistencia de los componentes de negocio ante recuperaciones. En caso de fallas de algún componente, no debe haber pérdida de información.
- ✓ Debe contemplar requerimientos de consistencia transaccional. Ante la falla del aplicativo, se debe contar con mecanismos que contemplen la interrupción de transacciones para que estas finalicen de manera correcta.

5.3.3 Seguridad

- ✓ La solución debe reflejar patrones de seguridad teniendo en cuenta la alta sensibilidad de la información que maneja de acuerdo a las especificaciones funcionales dadas y a las políticas, normas y estándares de seguridad requeridas.

5.3.4 Mantenibilidad

- ✓ Se debe estructurar el código de una manera consistente y predecible.
- ✓ Para objetos que son frecuentemente manejados en la lógica del negocio, implementar las respectivas interfaces que aseguren su fácil implementación en el sistema.
- ✓ Asegurar que el diseño de las interfaces contemple el que las propiedades públicas y los parámetros de los métodos sean de un tipo estandarizado.
- ✓ El sistema debe ser construido e implantado de tal manera que un cambio en los parámetros de negocio no obligue a la generación de una nueva versión del módulo.

Los requerimientos no funcionales son básicos dentro de un desarrollo de software porque son requerimientos que se encuentran implícitos dentro del mismo y que son totalmente imperceptibles para el cliente, son de suma importancia para las pruebas porque sin estos no se puede asegurar la verdadera funcionalidad y nivel de calidad del sistema.

Existe una lista infinita de posibilidades para probar estos requerimientos no funcionales pero dentro de las más importantes se encuentran:

Desempeño; es fácilmente cuantificable y validable, este requerimiento establece una métrica que puede ser la cantidad de trabajos que debe realizar por unidad de tiempo, o los plazos que se deben cumplir en una aplicación.

Escalabilidad; nos indica qué tan bien trabajará la solución a algún problema cuando el tamaño del problema se incremente, por carga solicitada, conexión simultánea, distribución, etc.

Modificabilidad; es una medida de qué tan fácil podría ser cambiar una aplicación para cumplir nuevos requerimientos funcionales y no funcionales. Predecir la modificabilidad requiere un estimativo del esfuerzo y/o costo de hacer un cambio.

Disponibilidad; relaciona con la confiabilidad de una aplicación, si una aplicación no está disponible para su uso cuando se necesita, entonces es improbable que cumpla sus requerimientos no funcionales, es fácil de especificar y medir.

5.4 REQUERIMIENTOS TÉCNICOS

Esta etapa tiene como objeto identificar el perfil tecnológico para tratar de conseguir la mayor adecuación tras la etapa de definición, una vez establecidos los requerimientos funcionales que ha de cumplir. En definitiva, de lo que se trata es de establecer cuáles deben ser los requisitos técnicos que debe presentar la aplicación, de acuerdo con el perfil tecnológico y los recursos disponibles, encaje dentro de la estrategia de los sistemas de información que se haya marcado. Para llevar a cabo esto es necesario realizar un análisis de los sistemas informáticos existentes. Si la evaluación técnica de los sistemas actuales ha sido positiva, el siguiente paso es determinar los requisitos técnicos que deben cumplir las aplicaciones y la definición concreta del hardware necesario.

Los requerimientos tecnológicos están definidos también por las especificaciones que necesita el lenguaje de programación donde se desarrolló el sistema de software, que incluye tanto especificaciones de hardware como en software, mínimas para el buen funcionamiento del sistema. Son de vital importancia porque sin estos no podría marchar el sistema de software desarrollado, por lo tanto no podría cumplir con los demás requerimientos necesarios para establecer la calidad del producto y cumplir con la satisfacción del cliente.

La forma más fácil de probar es por el medio visual ingresando en la máquina, validando y verificando su correcta instalación del hardware y software respectivamente.

En lo que a los sistemas de información con soporte informático se refiere, se tiene además que hablar del concepto de Arquitectura y, dentro de ésta, distinguir las dos vertientes que se presentan. La relacionada con los medios físicos y que constituyen el hardware de cada sistema y el orientado a la explotación de los medios y adaptado a las necesidades concretas de cada circunstancia.

La arquitectura de sistemas de información está caracteriza por dos aspectos básicos:

- ✓ El conjunto de principios y normas que tienen por objeto definir la interrelación entre sistemas inteligentes a efectos de las comunicaciones entre ellos.
- ✓ La definición y distribución de los elementos físicos y lógicos de los sistemas incluidos.

5.5 REQUERIMIENTOS LEGALES

Los requisitos legales que debe cumplir el sistema, es decir, toda la normativa legal que aplica al sistema, las restricciones legales de su uso y las normativas de gestión

de la información confidencial. También se incluyen los requisitos de destrucción de información confidencial al final del ciclo de vida de la misma.

Como principales objetivos podemos indicar los siguientes:

- ✓ Controlar que todas las actividades se realicen cumpliendo los procedimientos y normas fijados, evaluar su bondad y asegurarse del cumplimiento de las normas legales.
- ✓ Asesorar sobre el conocimiento de las normas.
- ✓ Definir, implantar y ejecutar mecanismos y controles para comprobar el logro de los grados adecuados del servicio informático, lo cual no debe considerarse como que la implantación de los mecanismos de medida y responsabilidad del logro de esos niveles se ubique exclusivamente en la función de control interno, si no que cada responsable de objetivos y recursos es responsable de esos niveles, así como de la implantación de los medios de medida adecuados.

Los requerimientos legales son probados con las auditorias informáticas internas programadas donde se verifica que las licencias utilizadas se encuentren en vigencia y estén siendo utilizadas correctamente, accesos, la parte de hardware todo esto está controlado bajo la Norma de Seguridad ISO 27000.

Los requerimientos legales son de vital importancia para evitar ser blancos de espionaje, delincuencia, riesgos, debilidades, combatir la corrupción y fomentar

buenas prácticas. Este requerimiento también ayuda a cumplir con la eficiencia del sistema, mejora la capacidad de análisis y evaluación de la información, mejora las actividades de control, dota a la institución de la agilidad necesaria para alcanzar la excelencia.

Capítulo VI

6 EXTRACCIÓN

6.1 ANÁLISIS DE LAS PRINCIPALES CARACTERÍSTICAS DE MÉTODOS TESTING

Hoy en día, debido al aumento del tamaño y la complejidad del software, el proceso de prueba se ha convertido en una tarea vital dentro del desarrollo de cualquier sistema informático.

Un aspecto fundamental del proceso de prueba es evaluar la satisfacción de la especificación funcional del sistema o requisitos por parte del sistema construido. Para garantizar el nivel de calidad del sistema construido es necesario verificar la correcta y completa implantación de los requisitos establecidos en las etapas iníciales del desarrollo.

Para tener un mejor enfoque se detalla la siguiente matriz de priorización a partir del análisis de cada una de sus características, la ponderación utilizada en la matriz se basa en la toma de la experiencia la de las empresas encuestadas a la hora de utilizar un método de testing. , para poder identificar cual método de prueba hasta el momento es el más confiable.

Tabla 6: MATRIZ DE PRIORIZACION

	Pruebas de caja blanca	Pruebas de caja negra	Pruebas de integración	Pruebas de aceptación	TOTAL
Pruebas de caja blanca		4	2	1	7
Pruebas de caja negra	6		3	2	11
Pruebas de integración	8	7		4	19
Pruebas de aceptación	9	8	6		23

ELABORADA POR: TULIO NAVARRETE

Se puede concluir que tiene mayores resultados al momento de realizar pruebas de testing es el de PRUEBAS DE ACEPTACION, con menores incidentes de errores y la entrega de requerimientos se lo realiza con mayor calidad, de acuerdo a resultados esperados por el cliente.

6.2 METODO DE CAJA NEGRA

Es un método con enfoque principalmente funcional, se centra en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiende a su especificación. Por ello se denominan pruebas funcionales, y el probador 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 sin que esto signifique que no sean útiles en cualquier módulo del sistema.

Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea.

El problema con las pruebas de caja negra no suele estar en el número de funciones proporcionadas por el módulo (que siempre es un número muy limitado en diseños razonables); sino en los datos que se le pasan a estas funciones. El conjunto de datos posibles suele ser muy amplio (por ejemplo, un entero).

Los beneficios permiten identificar claramente las entradas, salidas, estudiar las relaciones que existen entre ellas, permitiendo así maximizar la eficiencia**,** se centran en lo que se espera de un módulo, concentra su atención en el dominio de la información.

Mediante las técnicas de prueba de la caja negra se derivan un conjunto de casos de prueba que satisfacen los siguientes criterios:

- Casos de prueba que reducen, en un coeficiente que es mayor que uno, el número de casos de prueba adicionales que se deben diseñar para alcanzar una prueba razonable,
- Casos de prueba que nos dicen algo sobre la presencia o ausencia de clases de errores asociados solamente con la prueba en particular que se encuentra disponible.

6.3 METODO DE CAJA BLANCA

Las pruebas de caja blanca (también conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. El probador escoge distintos valores de entrada para examinar cada uno de los posibles flujos de ejecución del programa y cerciorarse de que se devuelven los valores de salida adecuados.

Al estar basadas en una implementación concreta, si ésta se modifica, por regla general las pruebas también deberán rediseñarse.

Aunque las pruebas de caja blanca son aplicables a varios niveles, habitualmente se aplican a las unidades de software. Su cometido es comprobar los flujos de ejecución dentro de cada unidad (función, clase, módulo, etc.) pero también pueden testear los flujos entre unidades durante la integración, e incluso entre subsistemas, durante las pruebas de sistema.

A pesar de que este enfoque permite diseñar pruebas que cubran una amplia variedad de casos de prueba, podría pasar por alto partes incompletas de la especificación o requisitos faltantes, pese a garantizar la prueba exhaustiva de todos los flujos de ejecución del código analizado.

Las principales técnicas de diseño de pruebas de caja blanca son:

Pruebas de flujo de control

Pruebas de flujo de datos

Pruebas de bifurcación (branchtesting)

Pruebas de caminos básicos

Dentro de sus beneficios garantizan que se ejercitan al menos una vez todos los caminos independientes de cada módulo, se ejercitan todas las decisiones lógicas en sus caras verdaderas y falsas, ejecutan todos los bucles en sus límites y con sus límites operacionales, se ejercitan las estructuras de datos internas para asegurar su validez.

6.4 METODO DE PRUEBAS DE INTEGRACION

Las pruebas de integración se llevan a cabo durante la construcción del sistema, involucran a un número creciente de módulos y terminan probando el sistema como conjunto.

Estas pruebas se pueden plantear desde un punto de vista estructural o funcional.

Las pruebas estructurales de integración son similares a las pruebas de caja blanca; pero trabajan a un nivel conceptual superior. En lugar de referirnos a sentencias del lenguaje, nos referiremos a llamadas entre módulos. Se trata pues de identificar todos los posibles esquemas de llamadas y ejercitarlos para lograr una buena cobertura de segmentos o de ramas.

Las pruebas funcionales de integración son similares a las pruebas de caja negra. Aquí trataremos de encontrar fallos en la respuesta de un módulo cuando su operación depende de los servicios prestados por otro(s) módulo(s). Según nos vamos acercando al sistema total, estas pruebas se van basando más y más en la especificación de requisitos del usuario.

Las pruebas finales de integración cubren todo el sistema y pretenden cubrir plenamente la especificación de requisitos del usuario. Además, a estas alturas ya suele estar disponible el manual de usuario, que también se utiliza para realizar pruebas hasta lograr una cobertura aceptable.

En todas estas pruebas funcionales se siguen utilizando las técnicas de partición en clases de equivalencia y análisis de casos límite (fronteras).

Los Beneficios atacan directamente a la implementación de los módulos desde el programa principal hacia los módulos inferiores de manera jerárquica, la selección de un tipo de prueba de integración dependerá directamente del software y a veces del plan del proyecto. En general el mejor compromiso puede ser una aproximación combinada que use la descendente para los niveles superiores de la estructura del programa junto con la ascendente para los niveles subordinados.

6.5 METODO DE PRUEBAS DE ACEPTACION

Estas pruebas las realiza el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable de cara al cliente; sino una vez pasada todas las pruebas de integración por parte del desarrollador.

La experiencia muestra que aún después del más cuidadoso proceso de pruebas por parte del desarrollador, quedan una serie de errores que sólo aparecen cuando el cliente se pone a usarlo.

Por esta razón, muchos desarrolladores ejercitan unas técnicas denominadas "pruebas alfa" y "pruebas beta". Las pruebas alfa consisten en invitar al cliente a que venga al entorno de desarrollo a probar el sistema. Se trabaja en un entorno controlado y el cliente siempre tiene un experto a mano para ayudarle a usar el sistema y para analizar los resultados.

Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno del cliente, un entorno que está fuera de control. Aquí el cliente se queda a solas con el producto y trata de encontrarle fallos (reales o imaginarios) de los que informa al desarrollador.

Las pruebas alfa y beta son habituales en productos que se van a vender a muchos clientes.

Los beneficios principales y por la razón que es preferida por las organizaciones como el método más eficaz utilizado hasta el momento es porque se tiene participación activa del usuario, que debe ejecutar los casos de prueba ayudado por miembros del equipo de pruebas. Están enfocadas a probar los requisitos de usuario, comprobar si el cliente está satisfecho con el producto desarrollado y si este producto cumple con sus expectativas.

Capítulo VII

7 ESTUDIO DE MERCADO

7.1 DISEÑO DE LA ENCUESTA

La encuesta está basada para dos sectores empresariales que tiene el Ecuador: Sector Público y Sector Privado, además sobre los servicios que brinda cada empresa, sobre estos parámetros iníciales se formuló la siguiente encuesta:

Nombre de la Empresa _____

1.- Tipo de empresa a: **Pública** ☐ **Privada** ☐ b: **PYMES** ☐ **Grande** ☐		**2.- Objetivo de la empresa** **Servicios** ☐ **Industrial** ☐ **Educativa** ☐ **TIC** ☐ **Comercio** ☐ **Turismo** ☐ **Banca** ☐ **Otros** ☐ _____
3.- La empresa dispone del Área de Computación (TIC – Tecnología de la Información y Comunicaciones) que brinda servicios informáticos y/o tecnológicos internos? **Si** ☐ **No** ☐		4.- Que estrategia utiliza la empresa para contar con los servicios informáticos o tecnológicos Todos los servicios son internos ☐ Todos los servicios son externos ☐ Parte de los servicios son internos y parte externos ☐ Otra estrategia ☐ _____

5.- La empresa, que servicios externos de Tecnología o de informática ha contratado?

Cursos de capacitación	
Desarrollo de Software	
Mantenimiento Equipos	
Asesoría para el mejoramiento de procesos	
Instalación y mantenimiento de redes de computación	
Otros	

_____

_____

6.- Su empresa desarrolla software?

Si totalmente	
Si parcialmente	
No	

7.- Tiene personal especializado en control de calidad y pruebas de certificación de software?

Si	
No	

8.- Si la respuesta de la pregunta 7 es Si, que perfil tienen las personas que realizan las pruebas de certificación del software?

Ingenieros en Sistemas	
Ingenieros de Procesos	
Técnicos o Tecnól. Informáticos	
Dueños de las aplicaciones (Funcionales)	
Otros	

_____

_____

9.- Si la respuesta de la pregunta 7 es Si, qué método de Pruebas o Testing de software utiliza?

Caja Blanca	
Caja Negra	
Caja Gris	
Buenas prácticas de ITIL	
Otras	

_____

_____

10.- En su empresa, en el Área TI

12.- En su empresa participan los usuarios finales

controlan el versionamiento de las aplicaciones en producción? **Si** ☐ **No** ☐ **11.- En su empresa, en el Área TI existe un Comité o un grupo de personas que aprueben los controles de cambio de las aplicaciones?** **Si** ☐ **No** ☐		**o dueños de las aplicaciones en el proceso de pruebas de certificación de software?** **Si totalmente** ☐ **Si parcialmente** ☐ **No** ☐ **13**.- **En su empresa, en el Área TI cuentan con ambientes completos para realizar las pruebas de certificación de software?** **Si totalmente** ☐ **Si parcialmente** ☐ **No** ☐
14.- Cuáles son las incidencias o inconvenientes más comunes cuando se pone en producción un aplicación nueva o actualizada **Migración de datos** ☐ **Funcionalidad de la aplicación** ☐ **Inconsistencia de datos** ☐ **Consultas, reportes, estadística** ☐ **Otros inconvenientes** ☐ _____		

7.2 RECOLECCION DE DATOS

La recolección de datos se lo realizó aplicando la encuesta diseñada para el efecto, luego se codificó cada variable de las preguntas, para crear la base de datos en la herramienta SPSS[18].

Se adjunta en la parte de Anexos.

7.3 PROCESAMIENTO DE DATOS

El procesamiento de los datos se lo hizo con la ayuda de la herramienta SPSS.

Se adjunta en la parte de Anexos.

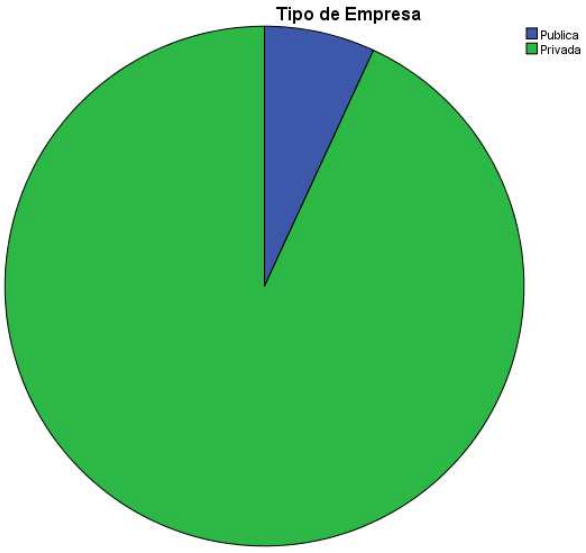
7.4 ANALISIS E INTERPRETACION DE DATOS

De acuerdo a la recolección y procesamiento de los datos se llegó a las siguientes conclusiones, para cada pregunta formulada en la encuesta:

[18] SPSS: programa estadístico usado en las ciencias sociales y las empresas de investigación de mercado.

Tipo de Empresa

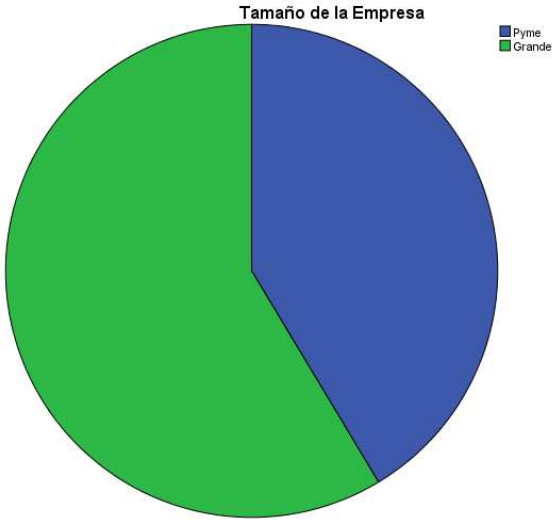
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	**Publica**	2	6,9	6,9	6,9
	Privada	27	93,1	93,1	100,0
	Total	29	100,0	100,0	



El 93.1% de las empresas encuestadas son empresas privadas, debido que se tuvo mayor apertura por parte de estas para realizar la encuesta.

Tamaño de la Empresa

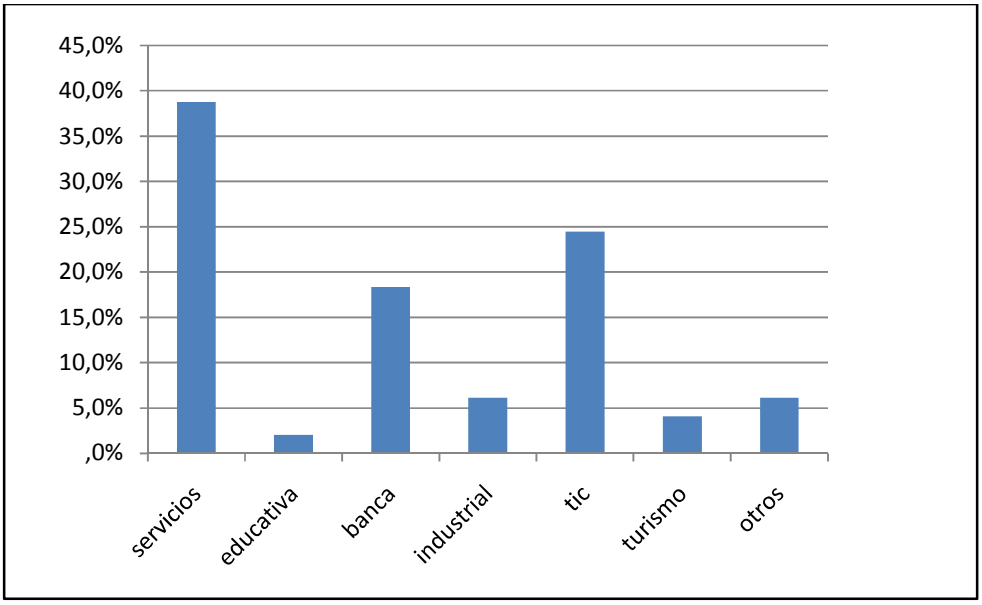
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado	MEDIA
Válidos	**Pyme**	12	41,4	41,4	41,4	14.5
	Grande	17	58,6	58,6	100,0	
	Total	29	100,0	100,0		



El 41.4 % de las empresas encuestadas son PYMES o se catalogan así por ser empresas pequeñas o medianas, donde también se incluyen a las microempresas, y el 58.6% son grandes, con una media total de 14.5 lo que nos puede indicar que en el país está existiendo un crecimiento rápido de empresas PYMES, lo que ayuda mucho a la estructura productiva y a variar según las necesidades del mercado

Objetivo de la Empresa

		Respuestas		Porcentaje de casos
		Nº	Porcentaje	
El objetivo de la empresa	servicios	19	38,8%	65,5%
	educativa	1	2,0%	3,4%
	banca	9	18,4%	31,0%
	industrial	3	6,1%	10,3%
	tic	12	24,5%	41,4%
	turismo	2	4,1%	6,9%
	otros	3	6,1%	10,3%
Total		49	100,0%	169,0%

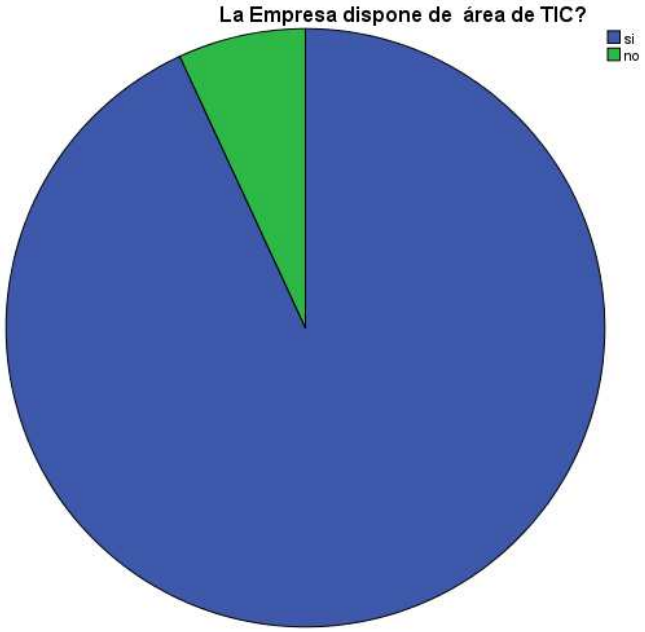


La mayor parte de las empresas encuestas (38,8%) son empresas de servicios, el siguiente porcentaje (24,5%) son empresas de TIC (Tecnologías de la Información y

Comunicación), y en tercer lugar empresas dedicadas a la banca (18,4%), son el tipo de empresas que actualmente están dominando el mercado ecuatoriano con implementación de grandes sistemas de software, para convertirse en pioneras de tecnología

La empresa dispone de área de TIC

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	**si**	27	93,1	93,1	93,1
	no	2	6,9	6,9	100,0
	Total	29	100,0	100,0	



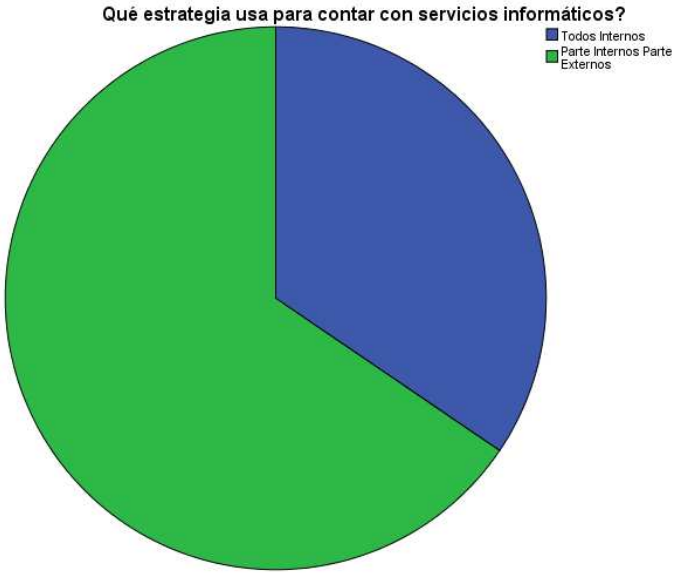
El 97% de las empresas encuestadas cuentan con un área de TIC, debido que actualmente las empresas se han dado cuenta de la importancia de tener un

departamento dedicado para controlar y monitorear sistemas de software, así como la implementación de nuevas soluciones según las necesidades y crecimiento de la empresa.

La alta gerencia ha visto la necesidad de invertir en tecnología para así lograr ser más competitivos en el mercado nacional y sobre todo abrir campo internacionalmente.

Qué estrategia usa para contar con servicios informáticos?

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Todos Internos	10	34,5	34,5	34,5
	Parte Internos Parte Externos	19	65,5	65,5	100,0
	Total	29	100,0	100,0	

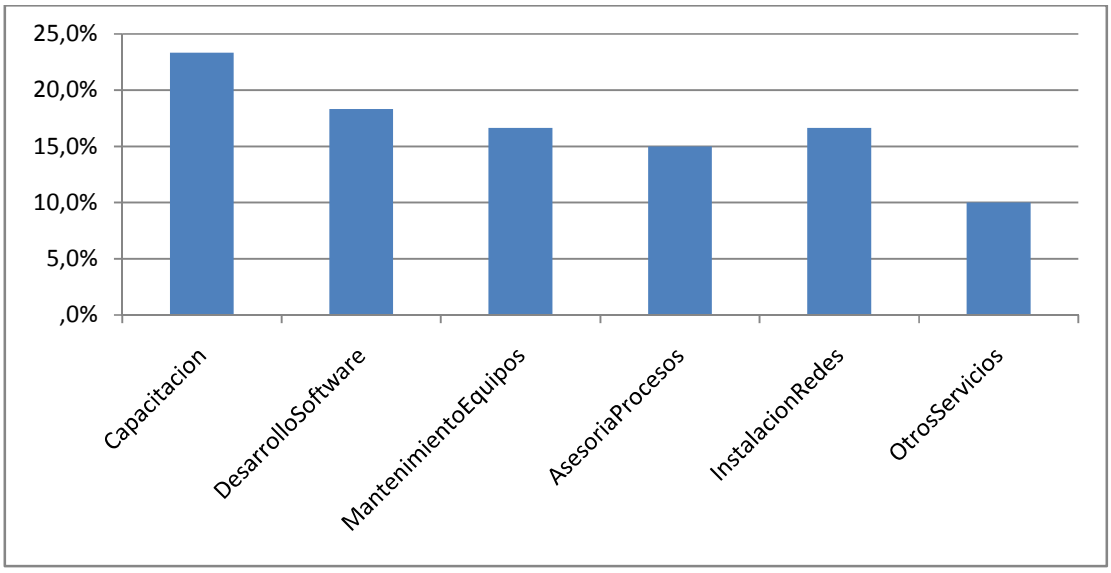


La mayoría de las empresas (65,5%) utilizan una técnica compartida de servicios de TIC, es decir parte de los servicios son internos y parte son contratados por fuera, debido que si bien Ecuador cuenta con la mano de obra capacitada para crear software, pero el mundo cambiante de la tecnología va mas rápido internacionalmente que ha provocado depender de ciertas herramientas y metodologías que son muchas veces contratados por fuera.

Qué Servicios externos ha contratado?

		Respuestas		Porcentaje de casos
		Nº	Porcentaje	
Qué servicio contrata	Capacitación	14	23,3%	48,3%
	Desarrollo Software	11	18,3%	37,9%
	Mantenimiento Equipos	10	16,7%	34,5%

	Asesoría Procesos	9	15,0%	31,0%
	Instalación Redes	10	16,7%	34,5%
	Otros Servicios	6	10,0%	20,7%
Total		60	100,0%	206,9%

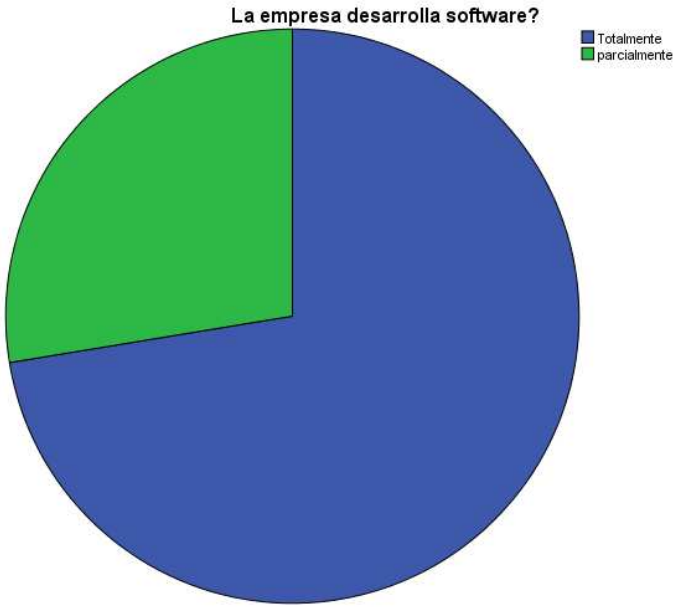


La mayor parte de las empresas contratan servicios de capacitación, porque desarrollan su propio software. Ya que tan solo el 18,3% de las empresas declaran contratar desarrollo de software por fuera.

La capacitación es una parte fundamental dentro del área TIC , muy necesaria para que los recursos tenga la capacidad de crear nuevo software, esa capacitación casi siempre debe ser exterior debido que las grandes herramientas de software utilizadas ha sido creada, y actualizadas por expertos internacionales con gran reconocimiento.

La empresa desarrolla software?

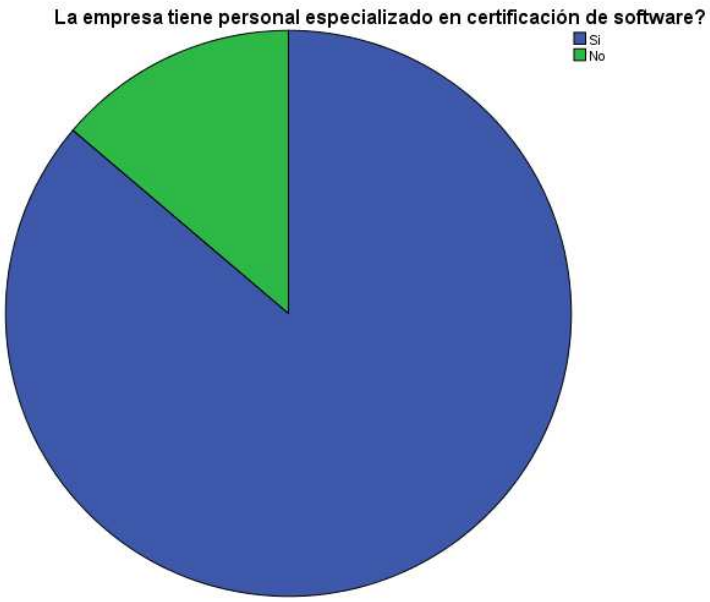
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Totalmente	21	72,4	72,4	72,4
	parcialmente	8	27,6	27,6	100,0
	Total	29	100,0	100,0	



El 72,4% de las empresas desarrollan software para sí mismo o para terceros según el objetivo de la empresa, para Ecuador es un gran avance para la parte tecnológica contar con empresas dedicadas a desarrollos que en un futuro podrían crear software que pueda ser reconocido internacionalmente, y puede abrir nuevos mercados y escalar en posicionamiento de países que crean tecnología.

La empresa tiene personal especializado en certificación de software?

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	**Si**	25	86,2	86,2	86,2
	No	4	13,8	13,8	100,0
	Total	29	100,0	100,0	

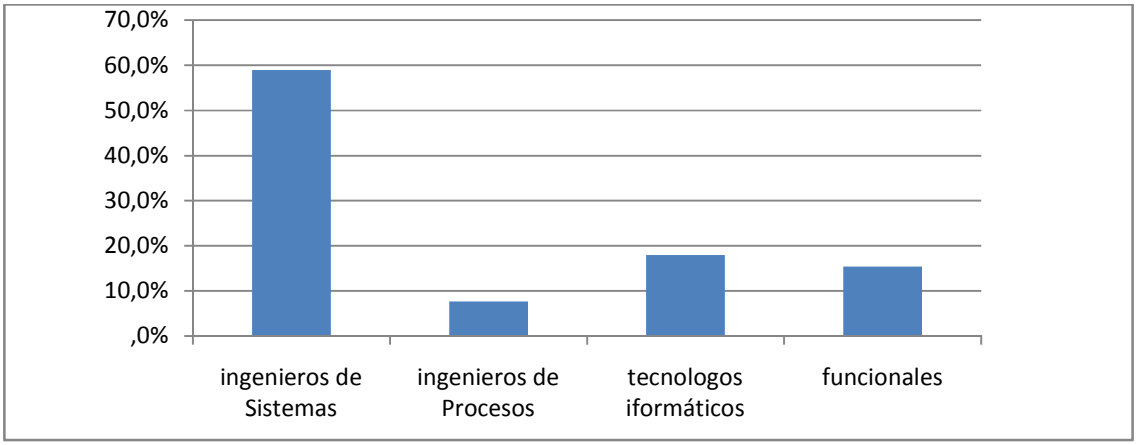


El 86,2% de las empresas encuestadas aseguran tener personal capacitado para la certificación del software, lo que indica que Ecuador tiene un alto porcentaje de personas bien capacitadas y la educación en tecnología cada vez está siendo más solida en el país, logrando ser muy competitivos a nivel internacional, lastimosamente actualmente el mercado nacional e internacional está muy saturado

de personas capacitadas para el desarrollo de software lo que valdría la pena es empezar a especializarlas en herramientas puntuales.

Qué perfil tiene el personal de certificación de software?

		Respuestas		Porcentaje de casos
		Nº	Porcentaje	
Perfil del personal de certificación	ingenieros de Sistemas	23	59,0%	92,0%
	ingenieros de Procesos	3	7,7%	12,0%
	tecnólogos informáticos	7	17,9%	28,0%
	funcionales	6	15,4%	24,0%
Total		39	100,0%	156,0%

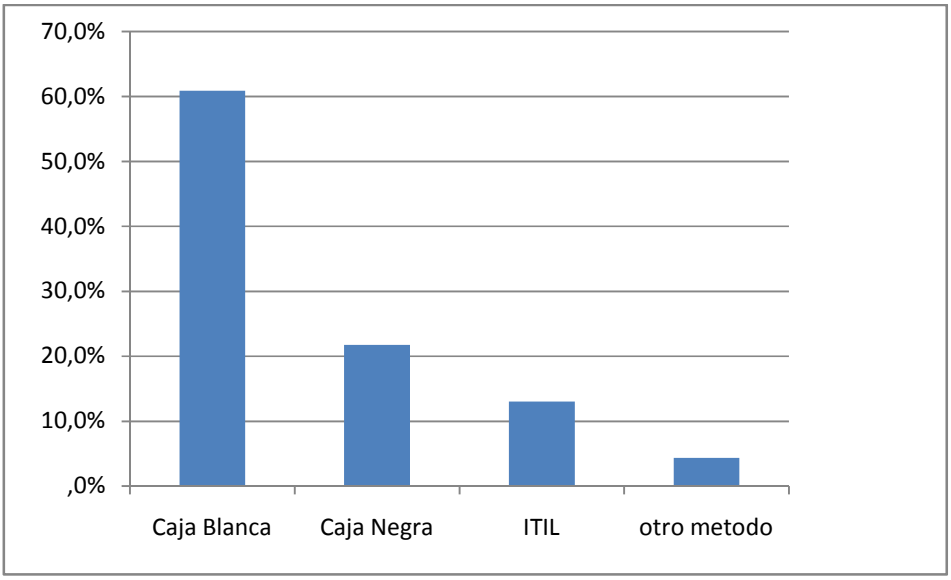


El 59% de las empresas cuentan con ingenieros de sistemas para la certificación del software que desarrollan y tan solo el 15,4% tienen funcionales para este objetivo, lo que indica que si bien se tiene expertos para la certificación del software , pero no existe personal suficiente capacitado para certificar la calidad del software , lo cual

puede bajar significativamente la posibilidad de abrir nuevos mercados, debido que actualmente el cliente es muy estricto con sus requerimientos y con la calidad de entrega de los mismos.

Qué método de testing de software utiliza?

		Respuestas		Porcentaje de casos
		Nº	Porcentaje	
Método de testing de software	Caja Blanca	14	60,9%	70,0%
	Caja Negra	5	21,7%	25,0%
	ITIL	3	13,0%	15,0%
	otro método	1	4,3%	5,0%
Total		23	100,0%	115,0%

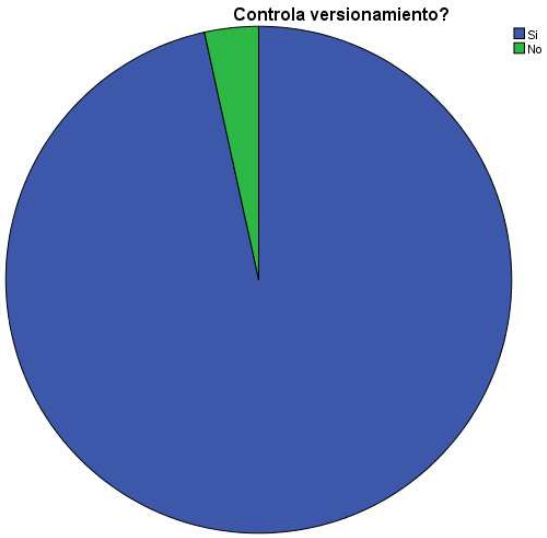


La mayoría de las empresas (60,9%) usa como método de testing Caja Blanca, lo que significa que el método utilizado si bien cubre una amplia variedad de casos de

prueba, podría pasar por alto partes incompletas de la especificación o requisitos faltantes, pese a garantizar la prueba exhaustiva de todos los flujos de ejecución del código analizado, esto baja la calidad del software y por consecuencia puede generar inconvenientes con el cliente.

Controla versionamientos

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	**Si**	28	96,6	96,6	96,6
	No	1	3,4	3,4	100,0
	Total	29	100,0	100,0	

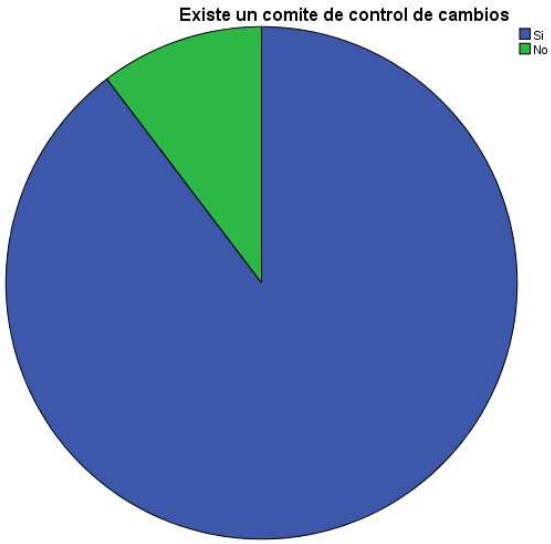


El 96,6 % de las empresas realiza versionamientos de software, lo cual es importante para tener un control minucioso de todos los objetos que se han utilizado o van

utilizarse dentro del desarrollo de software, la disciplina y la organización son un gran principio para acercase a brindar productos y servicios de alta calidad.

Existe un comité de control de cambios

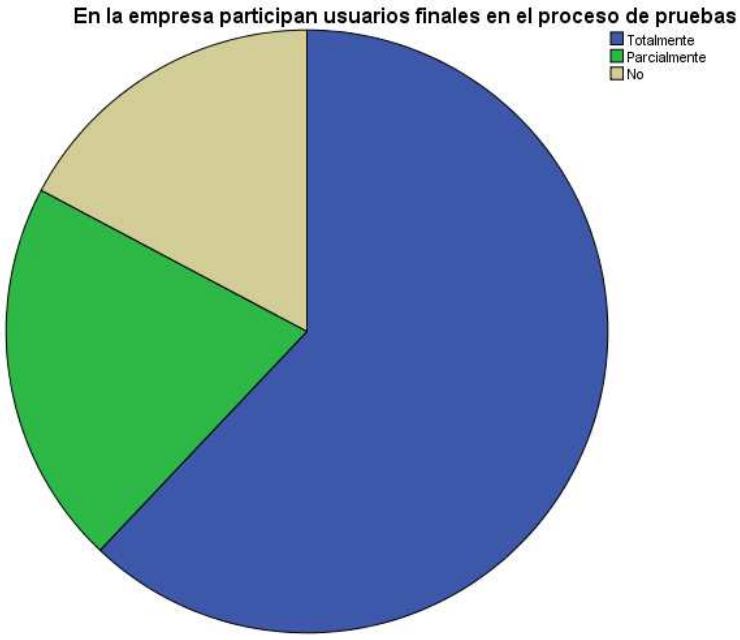
		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	**Si**	26	89,7	89,7	89,7
	No	3	10,3	10,3	100,0
	Total	29	100,0	100,0	



El 89,7 % de las empresas tienen un comité de control de cambios, lo cual indica que las organizaciones tienen claro lo importante de controlar cada versión del desarrollo de software, para no tener confusiones o retrasos en los mismos.

En la empresa participan usuarios finales en el proceso de pruebas

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Totalmente	18	62,1	62,1	62,1
	Parcialmente	6	20,7	20,7	82,8
	No	5	17,2	17,2	100,0
	Total	29	100,0	100,0	



El 62,1% de las empresas aseguran tener funcionales en el proceso de pruebas, lo cual es un buen comienzo tener un acercamiento con el cliente final, y para asegurarse que la entrega se encuentra dentro de los requerimientos especificados, pero sería importante que este porcentaje incremente desde el inicio del desarrollo para certificarse completamente de la calidad de software se está entregando.

La empresa cuenta con ambientes completos para realizar pruebas

		Frecuencia	Porcentaje	Porcentaje válido	Porcentaje acumulado
Válidos	Totalmente	21	72,4	72,4	72,4
	Parcialmente	7	24,1	24,1	96,6
	No	1	3,4	3,4	100,0
	Total	29	100,0	100,0	



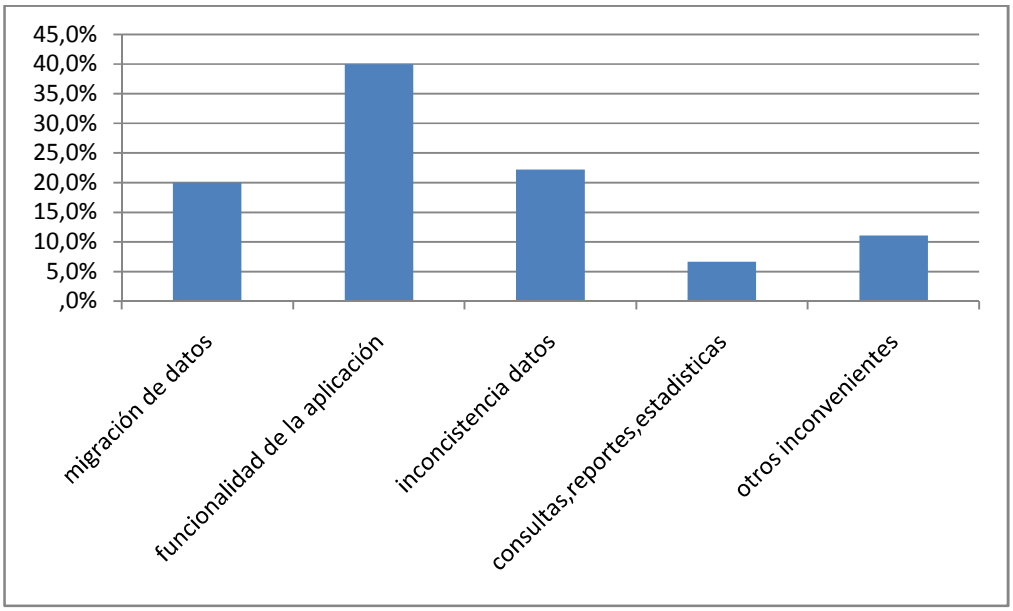
El 72,4% de las empresas dicen tener ambientes completos y similares a los ambientes de producción para realizar las pruebas, lo cual indica que se puede tener un proceso de pruebas similar a la realidad y tomar acciones correctivas y preventivas en caso de presentarse alguna inconformidad con el cliente, de forma

precisa y con la seguridad que dichos inconvenientes no estarán presentes en producción.

El 24,1 % de empresas que tienen ambientes parcialmente, deberían analizar la posibilidad de implementar ambientes parecidos para incrementar la calidad del software, y con esto posiblemente se reduciría gran parte de los problemas que actualmente presentan.

Cuáles son los inconvenientes más comunes cuando se pone en producción una aplicación

		Respuestas		Porcentaje de casos
		Nº	Porcentaje	
Cuáles son los inconvenientes comunes	migración de datos	9	20,0%	31,0%
	funcionalidad de la aplicación	18	40,0%	62,1%
	inconsistencia datos	10	22,2%	34,5%
	consultas,reportes,estadísticas	3	6,7%	10,3%
	otros inconvenientes	5	11,1%	17,2%
Total		45	100,0%	155,2%

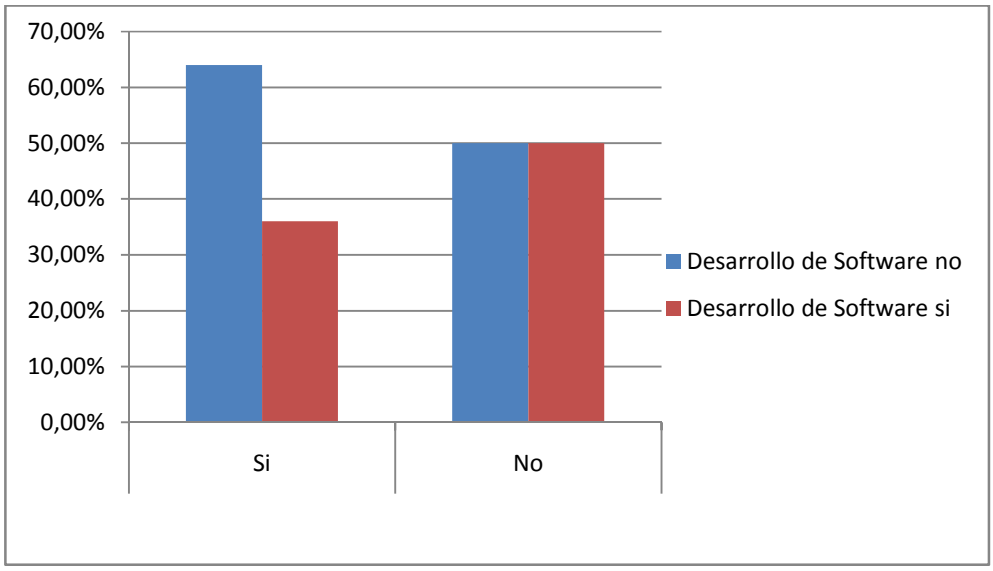


El mayor de los problemas que tienen las empresas en la puesta en producción es la funcionalidad de las aplicaciones (40%), el siguiente es inconsistencia (22,2%) de datos, problemas que se entenderían no deberían existir en base a las respuestas de la pregunta anterior donde la mayor parte de las empresas declaran tener ambientes similares a los de producción para el proceso de pruebas, debido que las pruebas funcionales no se realizan con el compromiso y la calidad que se debería, y la metodología empleada para realizar las pruebas tan eficiente como se esperaría.

VARIABLES CRUZADAS

Desarrollo de Software vs Personal de Q&A

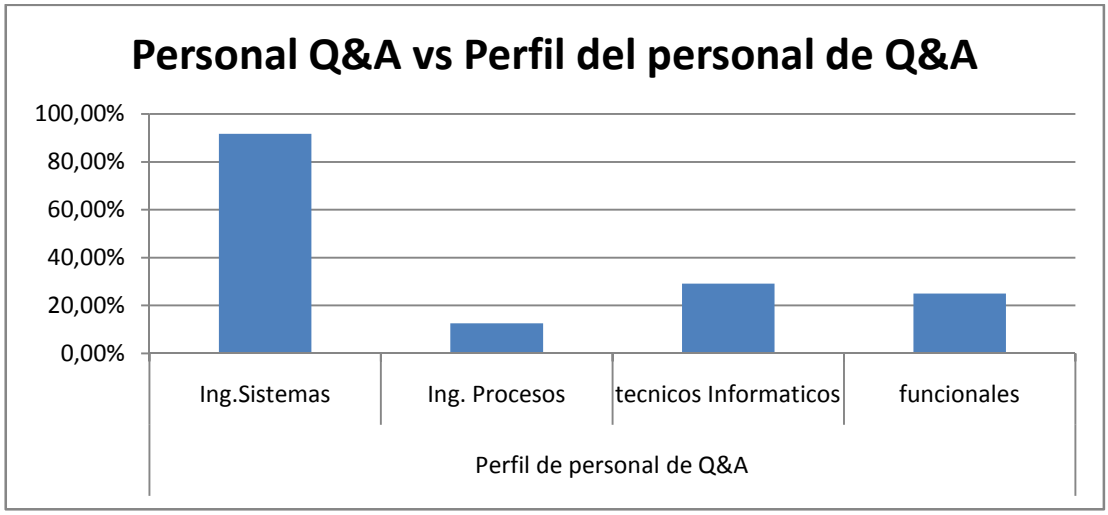
		Personal Q&A		Total
		Si	No	
Desarrollo de Software	no	64,00%	50,00%	62,10%
	si	36,00%	50,00%	37,90%



El 36% de las empresas que desarrollan software tienen personal de certificación. Lo interesante que las empresas que no realizan desarrollo de software tienen el 64% de personas más capacitadas, lo que indica claramente la calidad de software que actualmente las empresas de desarrollo de software realizan sin contar con la experiencia y educación necesaria de su personal, poniendo en juego mucho del prestigio de la empresa.

Personal de Q&A vs Perfil del Personal de Q&A

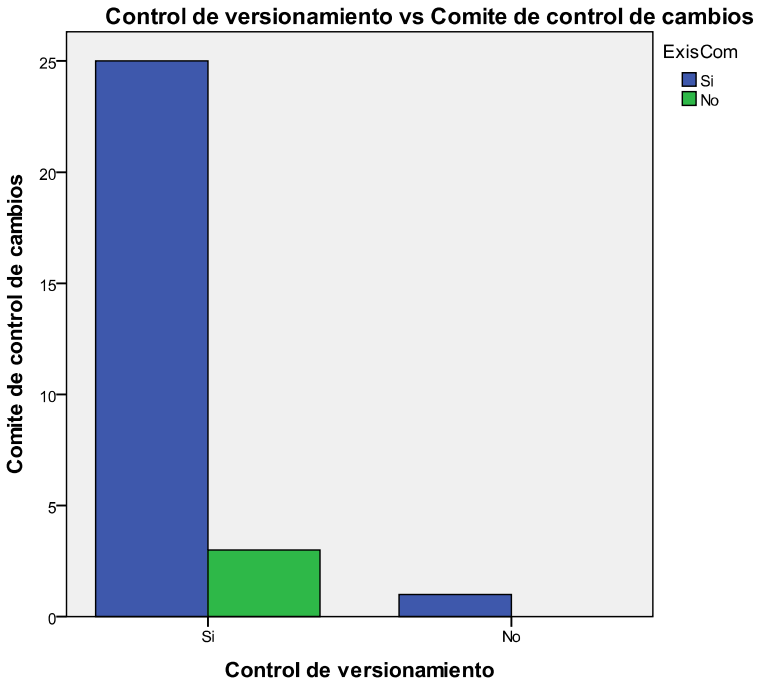
		Perfil de personal de Q&A			
		Ing. Sistemas	**Ing. Procesos**	**técnicos Informáticos**	**funcionales**
Personal Q&A	Si	91,70%	12,50%	29,20%	25,00%



El 91,7% del personal de Q&A en las empresas son profesionales en Ingeniería de Sistemas, y tan solo el 25% son funcionales o dueños de las aplicaciones, lo que indica que las pruebas son revisadas minuciosamente por un experto técnico, pero también debería tomarse en consideración en un experto funcional o el dueño de la aplicación para saber exactamente si la funcionalidad es la correcta y cumple con los requisitos especificados dados por el cliente.

Control de Versionamientos vs Comité de control de cambios

			ExisCom		Total
			Si	No	
Control de Versionamientos	Si	Recuento	25	3	28
		% dentro de ExisCom	96,20%	100,00%	96,60%
	No	Recuento	1	0	1
		% dentro de ExisCom	3,80%	0,00%	3,40%
Total		Recuento	26	3	29
		% dentro de ExisCom	100,00%	100,00%	100,00%

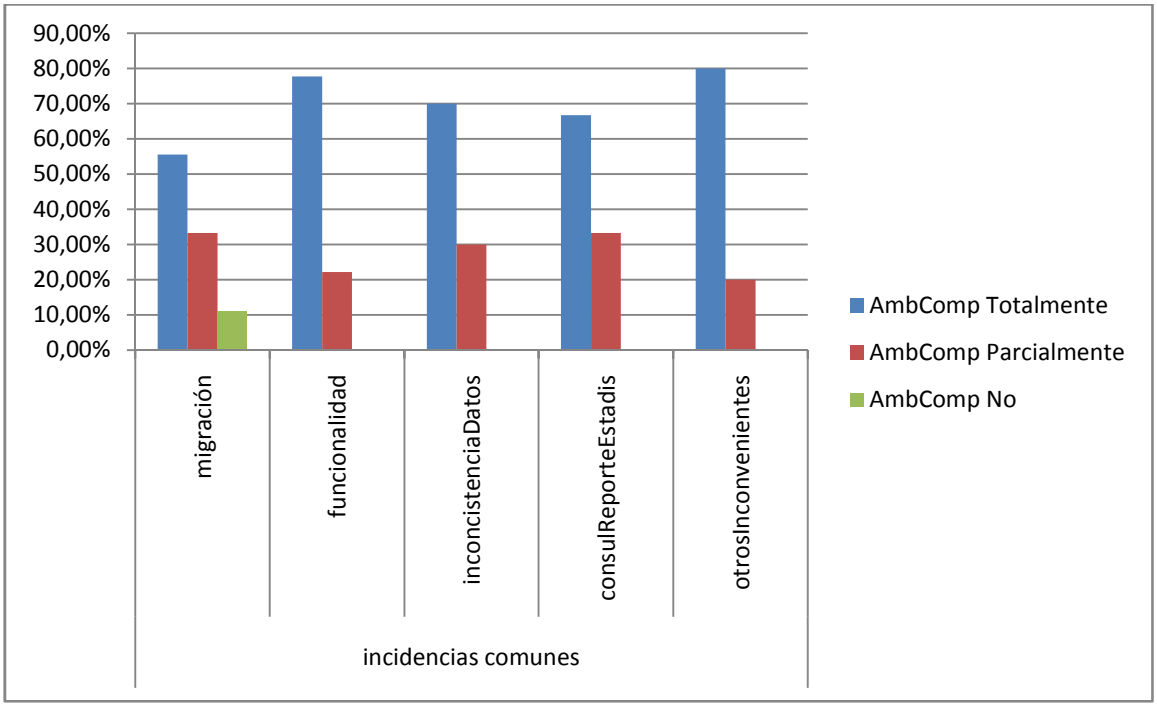


De las 26 empresas que afirman tener un comité de cambios, el 96,20% afirman tener un control de versionamientos de software, por otro lado de las 3 empresas que no tienen comité de cambios el 100% de estas afirma tener control de

versionamientos. Lo que nos indica claramente que actualmente las empresas si tienen la buena práctica de tener un control de los versionamientos, pero no la buena práctica de un control de los cambios, los cual generan un problema al momento de realizar pruebas de testing.

Se tienen ambientes completos para realizar pruebas VS Incidencias comunes

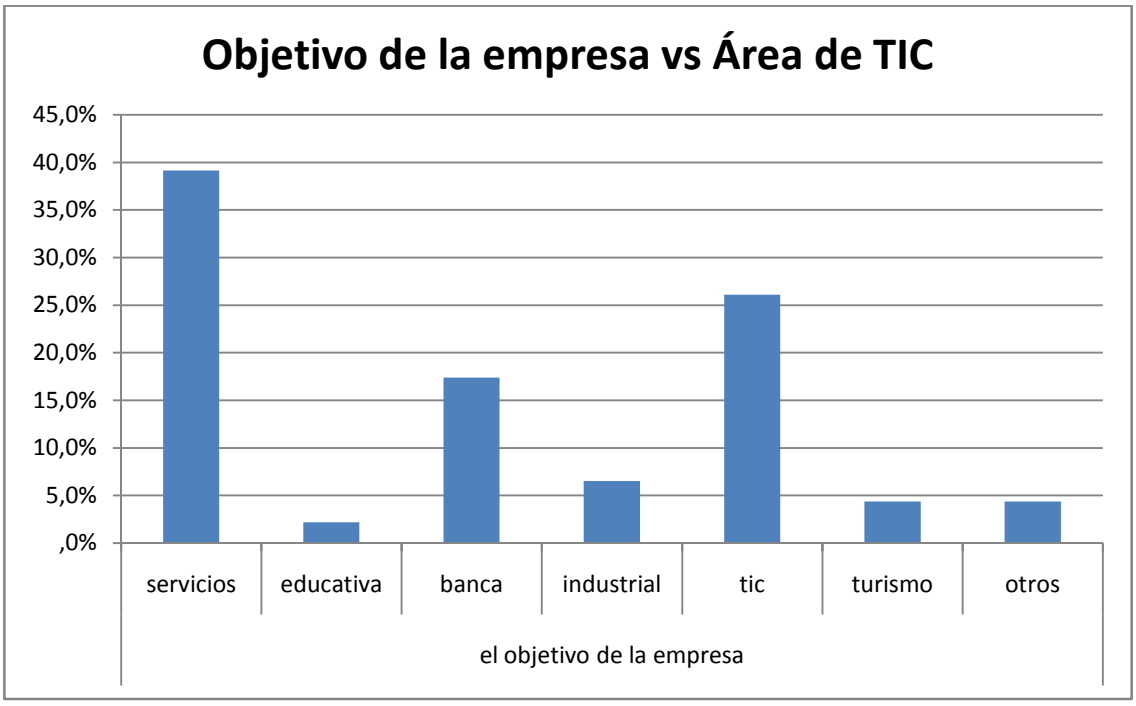
		incidencias comunes					
		migración	funcionalidad	inconcistenciaDatos	consulReporteEstadis	otrosInconvenientes	Total
AmbComp	Totalmente	55,60%	77,80%	70,00%	66,70%	80,00%	21
	Parcialmente	33,30%	22,20%	30,00%	33,30%	20,00%	7
	No	11,10%	0,00%	0,00%	0,00%	0,00%	1
Total							29



La funcionalidad es uno de los principales problemas que se reportan en la puesta en producción a pesar de tener ambientes de pruebas similares a los de producción. Esto se puede dar por varias razones, una de ella puede suceder porque a los desarrolladores no se los capacita bien sobre la funcionalidad y del negocio para que pueda enfocarse realmente a las necesidades del cliente, y por otra está el Ing. de pruebas desconoce también la parte del negocio y realiza pruebas, que no se acercan mucho a la realidad a pesar de tener ambientes idénticos a los de producción.

Objetivo de la empresa vs dispone de área de TIC

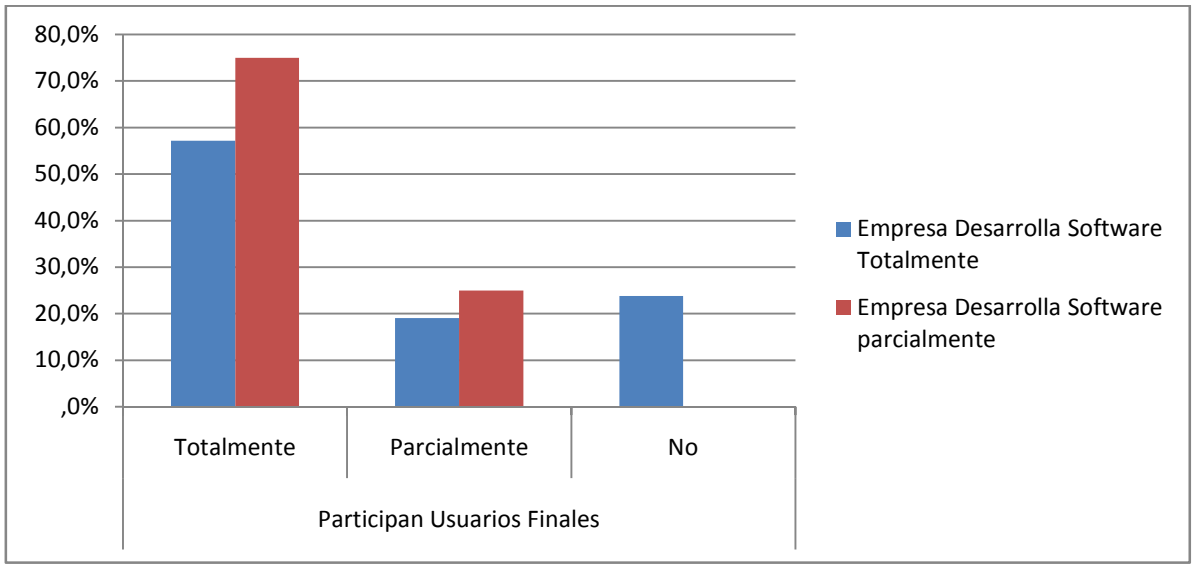
		area de TIC	Total
		si	
el objetivo de la empresa	servicios	39,1%	18
	educativa	2,2%	1
	banca	17,4%	8
	industrial	6,5%	3
	tic	26,1%	12
	turismo	4,3%	2
	otros	4,3%	2
Total			46



El 39,1% de las empresas de servicios poseen un área de TIC, el 26,1% de las empresas cuya naturaleza es TIC tienen un área especializada en TIC. Lo que significa que es importante para las empresas tener un área de TIC, pero la funcionalidad y el enfoque que tiene posiblemente está mal direccionado, es por tal motivo que se tiene una baja calidad a los desarrollos y pruebas realizadas.

Empresa Desarrolla Software VS Participan usuarios finales en certificación

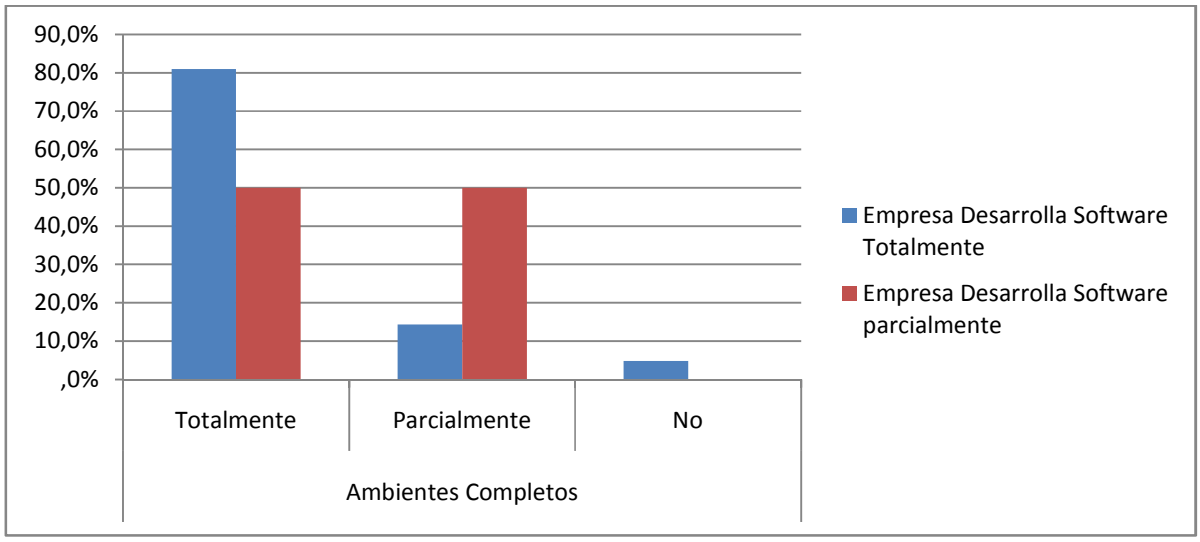
		Participación Usuarios Finales			Total
		Totalmente	Parcialmente	No	
Empresa Desarrolla Software	Totalmente	57,1%	19,0%	23,8%	100,0%
	parcialmente	75,0%	25,0%	,0%	100,0%



Tan solo el 57,1% de las empresas que desarrollan software hacen que los usuarios finales participen en el proceso de certificación de software. Lo cual se demuestra un bajo porcentaje y por lo tanto como consecuencia en la inconformidad del usuario final y de los clientes, porque las quejas y dudas no son solventadas a tiempo, y se lo realiza en ocasiones en vivo.

Empresa Desarrolla Software VS Cuenta con ambientes completos para pruebas

		Ambientes Completos			Total
		Totalmente	Parcialmente	No	
Empresa Desarrolla Software	Totalmente	81,0%	14,3%	4,8%	100,0%
	parcialmente	50,0%	50,0%	,0%	100,0%



El 81% de empresas que desarrollan software cuentan con un ambiente de pruebas similar al de producción. Y el 50% de las empresas que desarrollan software parcialmente tienen ambientes de prueba completos. Lo cual sería interesante identificar la causa raíz de los inconvenientes y la baja calidad del producto presentada al cliente.

Con base en los resultados obtenidos en la encuesta, se demuestra que la gran mayoría de las empresas dedicadas al desarrollo de software cuentan con equipos de testing para el desarrollo de sus proyectos, sin que esto asegure la calidad del producto final, ya que por varias razones como son, personal no calificado, pruebas en momentos inadecuados, exclusión del usuario final en el proceso de testing, ambientes totalmente diferentes a los de producción, etc. No se obtiene el resultado esperado de los productos finales.

Capítulo VIII

8 FORMULACION DE UN NUEVO METODO DE TESTING

De acuerdo al análisis de las características de los métodos más comunes, los resultados en las encuestas y las conclusiones obtenidas de las mismas, se pudo observar que las empresas actualmente tiene un déficit en las pruebas funcionales a pesar de tener un área dedicada para que esto no ocurra, lo que hace que el producto final tenga un porcentaje elevado de errores una vez que se ha puesto en producción, detalle que a su vez hace que él o los desarrolladores sigan tapando fisuras de programación con el fin de conseguir mejorar la funcionalidad del producto final que ya se encuentra en producción, actividades que implican costos de tiempo y económicos, debido a que el mantenimiento post entrega se convierta en una programación post entrega, con esto, se cae en el típico modelo prueba-error-corrección-prueba, de cada una de las falencias encontradas.

Es por lo tanto que se plantea este nuevo método para garantizar la calidad, y estabilidad al momento de entregar un software.

Planes de un control de versionamientos, control de cambios; con la simulación de ambientes de producción en test y con la participación de todos

los involucrados en el proyecto debiendo ser este personal capacitado para pruebas funcionales, procesos, técnico, y de QA.

8.1 Planificación Control de Versionamientos

Identificar todos los objetos involucrados.

Mantener una base de datos con históricos de petición de cambio con el usuario al que fue asignado.

Utilizar alguna herramienta de software que permita realizar el versionamientos de los objetos para mayor control.

Capacitación constante de las personas involucradas sobre el manejo de los objetos y en las personas que liberan dichos objetos.

8.2 Planificación Control de cambios.

Tener un cronograma de actividades para cada objeto a ser versionado.

Identificar claramente en qué ambiente va ha ser versionado, tomando las precauciones necesarias.

Informar a todo el personal involucrado sobre el versionamientos a realizarse para considerar colaterales.

Mantener un registro de los controles de cambios realizados.

8.3 Simulación de ambientes:

La simulación tiene como principal objetivo la predicción, es decir, puede mostrar lo que sucederá en un sistema real cuando se realicen determinados cambios bajo determinadas condiciones. Para garantizar la calidad de un desarrollo de software es necesario tener una simulación de ambientes de software de producción para realizar las pruebas técnicas y funcionales, con los usuarios expertos, esto permitirá corregir errores en momentos oportunos.

La simulación involucra dos facetas:

1) Construir el modelo

2) Ensayar diversas alternativas con el fin de elegir y adoptar la mejor en el sistema real, procurando que sea la óptima o que por lo menos sea lo suficientemente aproximada.

Se debe comenzar con un analisis previo de los requerimientos del usuario y del sistema , que pueden ser comprendidas en:

- Analizar requerimientos de usuario, de manera que especifique las preferencias y tareas.
- Especificar los requerimientos del sistema , utilizando cualquier diagrama de manera que describa el conjunto de servicios que ofrece el sistema al usuario.
- Proponer un prototipo de interfaz grafica.

- Asociar por cada interactor de la interfaz , la accion del usuario, el servicio que se ejecutara y el estado de interaccion.

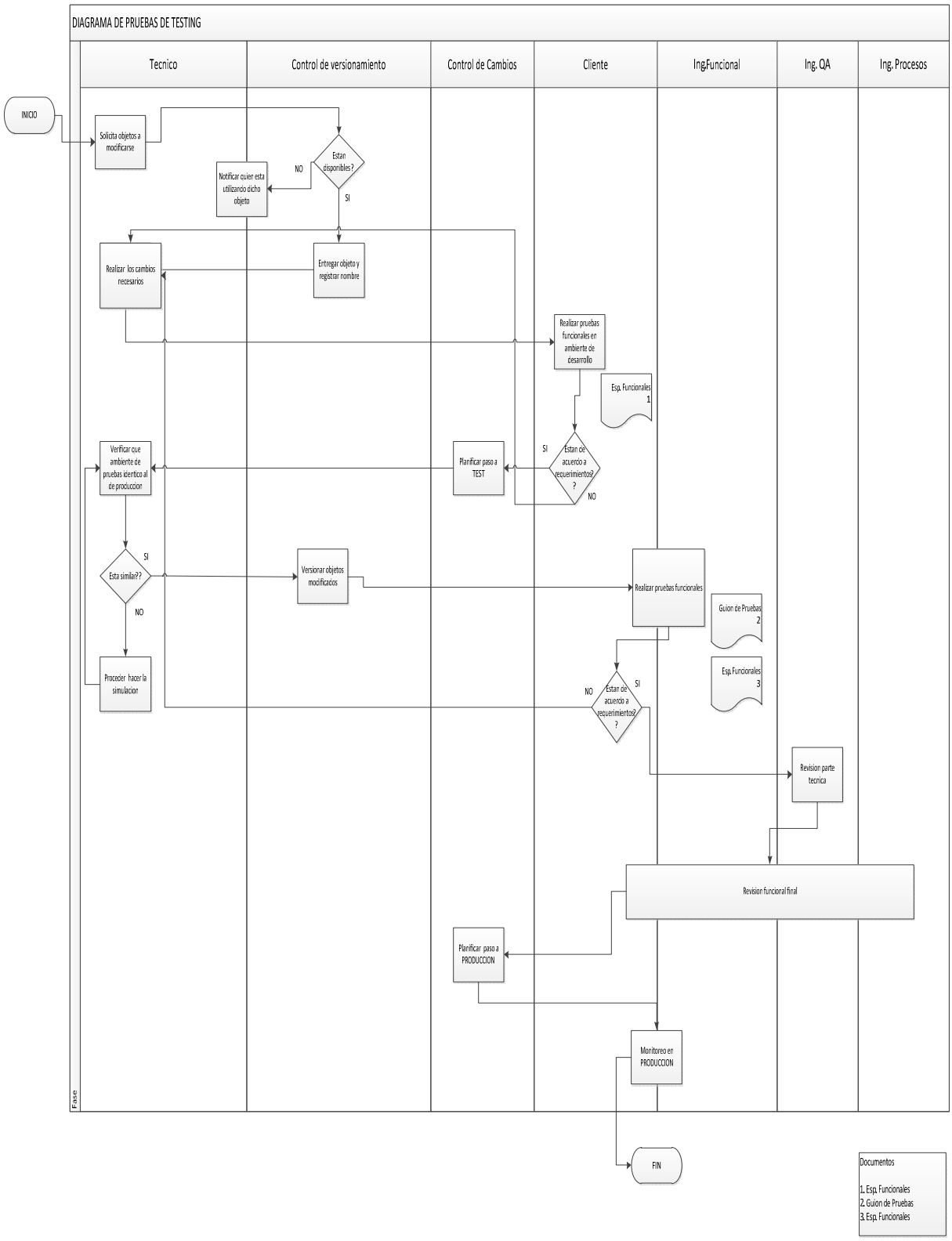
8.3.1 Ventajas:

- Adquirir una rápida experiencia a muy bajo costo y sin riesgos, no se compromete la confiabilidad del sistema.
- Identificar en un sistema complejo aquellas áreas con problema.
- Utilizarse en pruebas para los clientes, les permite probar sus medidas en el modelo, ver resultados luego de pasado el período simulado. Se repite el proceso durante varios períodos. Se analizan errores, se comparan estrategias hallando ventajas y desventajas de cada una.
- Puede ser aplicada para diseño de sistemas nuevos en los cuales se quieren comparar alternativas muy diversas surgidas de utilización de diferentes tecnologías, durante la vida de un sistema, para probar modificaciones antes que estas se implementen.

8.3.2 Personal Capacitado:

- Identificar al personal involucrado
- De acuerdo a la duración del proyecto y/o desarrollo de software, preparar capacitación al personal de las pruebas funcionales, del proceso, técnico, y de QA.
- Capacitación debe estar orientada al negocio al que va hacer referencia el software.
- En cada cambio del requerimiento del usuario debe ser notificado al personal.
- Mantener un registro de las capacitaciones, y debidamente documentadas.

8.3.3 Diagrama de Flujo de pruebas con el método propuesto.


DIAGRAMA DE PRUEBAS DE TESTING
Tecnico
Control de versionamiento
Control de Cambios
Cliente
Ing.Funcional
Ing. QA
Ing. Procesos
INICIO
Solicita objetos a mocificarse
Estan disponibles?
NO
SI
Notificar quien esta utilizando dicho objeto
Realizar los cambios necesarios
Entregar objeto y registrar nombre
Realizar pruebas funcionales en ambiente de desarrollo
Esp. Funcionales 1
Verificar que ambiente de pruebas identico al de produccion
Planificar paso a TEST
Estan de acuerdo a requerimientos? ?
SI
NO
Esta similar??
SI
NO
Versionar objetos modificados
Realizar pruebas funcionales
Guion de Pruebas 2
Esp. Funcionales 3
Proceder hacer la simulacion
Estan de acuerdo a requerimientos? ?
NO
SI
Revision parte tecnica
Revision funcional final
Planificar paso a PRODUCCION
Monitoreo en PRODUCCION
FIN
Fase
Documentos
1. Esp. Funcionales
2. Guion de Pruebas
3. Esp. Funcionales

Capítulo IX

9 PRUEBA DE TESTING DE SOFTWARE APLICANDO EL METODO PROPUESTO

9.1 DEMOSTRACION DEL MODELO

La demostración del modelo propuesto se realizó en la empresa Expertweb Cía. Ltda. Es una empresa especializada en el desarrollo e integración de sistemas con tecnología web, con más de seis años de experiencia en el mercado diseñando soluciones para los más diversos campos de acción: supervisión y control, telecomunicaciones, recursos humanos, banca y seguros, sitios web corporativos, catálogos dinámicos de productos, carros de compra, cotizadores en línea, manejadores de contenido, mercados virtuales, sistemas contables, sistemas financieros, entre otros.

Dentro de sus principales clientes se encuentran:

- Akros / ATA Solutions
- Royal Prestige Ecuador
- Grupo R&B Importadores
- Advantage Travel – Ecuador
- Habitat Ecuador - Grupo Inmobiliario Habitat

- Legislación Ecuatoriana - Libre Acceso
- Porfinempleo.com - MIDIAM S.A
- Clínica Odontológica Global Dental entre otros.

Para la demostración del modelo se la realizo en base a un proyecto llamado **Desarrollo de Software CSC para administración de casos de reparación para RIM** para la empresa Blackberry.

El proyecto consiste en: En el desarrollo del un software de administración de casos de reparación RIM, donde permitirá a HTM tener un nivel de atención apropiado para lograr una certificación por parte de la firma. Además con el fin de reparar los dispositivos Blackberry en nombre de RIM, el fabricante debe integrar sus sistemas de control de piso de talle rcon el sistema de RIM "s de seguimiento para ERMA.

La siguiente sección contiene una descripción simple de los pasos del proceso que intervienen en la creación de pedidos de los clientes en el sistema ERMA BlackBerry, incluyendo los pasos resaltados que requieren las comunicaciones B2B entre RIM y el Fondo de Reparación.

La creación de RMA con ERMA impone las siguientes restricciones:

- Todos los pedidos son de un tipo único modelo.

- Todos los pedidos son par a una oferta de servicios única(a granel, es decir de reparación Depoto DOA, etc.).
- Algunas órdenes pueden ser de un solo color.
- Cada orden debe ser recibido físicamente en el centro de la reparación como una orden completa(es decir, sin recepciones parciales) antes del evento de recibos se presenta.
- Actualizaciones de estado de triage debe presentarse en lotes y para un solo fin de RMA.
- Actualizaciones de los síntomas y la acción del estado debe ser presentada en los lotes y para un solo fin de RMA.
- Todas las actualizaciones de eventos para un equipo debe ser enviado en el orden correcto flujo de eventos.
- Eventos de síntomas y la acción no se puede actualizarla disposición final después de la disposición final y eventos de envío de la parte posterior centro de reparación para el cliente podrá incluir los traslados parciales en función de las preferencias del cliente.
- Los vendedores de reparación recibirán una orden de partido tercero de compra(PO)por el envío de RMA para la facturación de servicios de nuevo a RIM.

A continuación se presenta la planificación realizada antes de implementar el modelo propuesto:

Project Scope	4 días	jue 17/05/12	mar 22/05/12	
Analysis / Software requirements	23 días	jue 24/05/12	lun 25/06/12	
HTM Requirements	4 días	jue 24/05/12	mar 29/05/12	
RIM Requirements	4 días	mié 30/05/12	lun 04/06/12	3
Documentation / Reviews / Tunning	15 días	mar 05/06/12	lun 25/06/12	4
Database Design / Tune / 1st versior	7 días	mar 26/06/12	mié 04/07/12	5
Database Mapping	4 días	jue 05/07/12	mar 10/07/12	7
Software Layout Design	6 días	lun 18/06/12	lun 25/06/12	
Parameters Administration	6 días	mié 11/07/12	mié 18/07/12	8
1st Module: RMA Creation	15 días	lun 23/07/12	vie 10/08/12	
1st Module: test	5 días	lun 13/08/12	vie 17/08/12	13
RIM integration RMA Order Details	8 días	lun 20/08/12	mié 29/08/12	14
RMA Order Details test	10 días	jue 30/08/12	mié 12/09/12	15
2nd Module: Logistics	12 días	lun 20/08/12	mar 04/09/12	
2nd Module: test	5 días	mié 05/09/12	mar 11/09/12	18
RIM Integration: RMA Status Update	8 días	mié 12/09/12	vie 21/09/12	19
RMA Status Update test	5 días	lun 24/09/12	vie 28/09/12	20
3rd Module: Technical Dept.	10 días	mié 12/09/12	mar 25/09/12	
3rd Module: test	3 días	mié 26/09/12	vie 28/09/12	23
RIM Integration: RMA Status Update	4 días	lun 01/10/12	jue 04/10/12	24
RMA Status Update test	4 días	vie 05/10/12	mié 10/10/12	25

Se puede visualizar, que el tiempo asignado para test es muy poco y la corrección de las observaciones dadas también es pequeña; además de la definición de requerimientos es muy poca y puede existir ambigüedades.

A continuación se presenta la planificación con el modelo propuesto:

Project Scope	4 días	jue 17/05/12	mar 22/05/12	
Analysis / Software requirements	30 días	jue 24/05/12	mié 04/07/12	
HTM Requirements	10 días	jue 24/05/12	mié 06/06/12	
RIM Requirements	10 días	jue 07/06/12	mié 20/06/12	3
Documentation / Reviews / Tunning	10 días	jue 21/06/12	mié 04/07/12	4
Database Design / Tune / 1st versior	7 días	jue 05/07/12	vie 13/07/12	5
Database Mapping	5 días	lun 16/07/12	vie 20/07/12	7
Software Layout Design	10 días	lun 18/06/12	vie 29/06/12	
Parameters Administration	6 días	lun 23/07/12	lun 30/07/12	8
1st Module: RMA Creation	20 días	lun 23/07/12	vie 17/08/12	
1st Module : developing test	5 días	lun 20/08/12	vie 24/08/12	13
1st Module: developing corrections client	3 días	lun 27/08/12	mié 29/08/12	14
1st Module: second developing test	2 días	jue 30/08/12	vie 31/08/12	15
1st Module: test	5 días	lun 03/09/12	vie 07/09/12	16
RIM integration RMA Order Details	5 días	lun 10/09/12	vie 14/09/12	17
RMA Order Details test	3 días	lun 17/09/12	mié 19/09/12	18
2nd Module: Logistics	15 días	lun 20/08/12	vie 07/09/12	
2st Module : developing test	5 días	lun 10/09/12	vie 14/09/12	21
2st Module: developing corrections client	3 días	lun 17/09/12	mié 19/09/12	22
2st Module: second developing test	1 día	jue 20/09/12	jue 20/09/12	23
2nd Module: test	5 días	vie 21/09/12	jue 27/09/12	24
RIM Integration: RMA Status Update	3 días	vie 28/09/12	mar 02/10/12	25

3rd Module: Technical Dept.	15 días	mié 12/09/12	mar 02/10/12	
3st Module : developing test	8 días	mié 03/10/12	vie 12/10/12	29
3st Module: developing corrections client	5 días	lun 15/10/12	vie 19/10/12	30
3st Module: second developing test	3 días	lun 22/10/12	mié 24/10/12	31
3rd Module: test	3 días	jue 25/10/12	lun 29/10/12	32
RIM Integration: RMA Status Update	3 días	mar 30/10/12	jue 01/11/12	33
RMA Status Update test	2 días	vie 02/11/12	lun 05/11/12	34
4th Module: Billing	5 días	mar 06/11/12	lun 12/11/12	35
4st Module : developing test	3 días	mar 13/11/12	jue 15/11/12	37
4st Module: developing corrections client	2 días	vie 16/11/12	lun 19/11/12	38
4st Module: second developing test	1 día	mar 20/11/12	mar 20/11/12	39
4th Module: test	1 día	mié 21/11/12	mié 21/11/12	40
RIM Integration: Customer Vendor Inv	1 día	jue 22/11/12	jue 22/11/12	41
Customer Vendor Invocing test	1 día	vie 23/11/12	vie 23/11/12	42
RIM Integration: Exception Handling	1 día	lun 01/10/12	lun 01/10/12	

Donde se puede visualizar que se incrementó el número de días para el análisis de los requerimientos, se implementó el modelo propuesto al realizar las pruebas en ambiente de desarrollo con todo el personal involucrado, minimizando el tiempo en test y la posterior entrega en producción y de la integración.

Los tiempos iníciales se vieron desplazados de acuerdo a la planificación inicial, el aumento fue de 12 días, y se está garantizando al cliente que se entrega un producto con mayor calidad, y probado bajo su supervisión de acuerdo a los requerimientos obtenidos desde el inicio. Además se certifica que no existirán inconvenientes posteriores a la entrega.

Capítulo X

10 CONCLUSIONES Y RECOMENDACIONES

10.1 CONCLUSIONES

1. Cuando existe la apertura por parte de los directivos de las empresas para mejorar su operatividad, su rendimiento, en objetivo de brindar un servicio de calidad y/o entregar una herramienta de software ,realizando debidamente las pruebas necesarias de manera eficaz y efectiva, acogiendo métodos que pueden ir evolucionando de acuerdo al crecimiento que ofrece el mercado.

2. La aplicación de este método permitirá de manera fácil, rápida y sencilla validar los resultados del desarrollo de software de acuerdo a los requerimientos del cliente con la mayor calidad y con la aceptación adecuada.

3. El cliente se involucra directamente en cada fase a ser entregada, al ser probada en el simulador de ambiente conjuntamente con el personal, para corroborar que los requerimientos establecidos se cumplan y estén de acuerdo a los parámetros; tanto de manera funcional como técnica.

4. Se investigó a detalle la funcionalidad de cada uno de los métodos de testing , analizando ventajas y desventajas en su utilización,

5. Se realizó el análisis comparativo de las ventajas entre las diferentes de prueba de testing de software , mediante las encuestas realizadas a diferentes empresas desarrolladoras de software se pudo comprobar que usan varios métodos , pero no han logrado alcanzar los resultados esperados de entregar un producto con la más alta calidad de acuerdo a lo planificado.

6. Se propuso un nuevo modelo para cubrir cuellos de botellas en las pruebas de testing y sean solventadas las inquietudes y sugerencias del cliente antes del paso a test, para no tener demoras en pruebas y correcciones posteriores, sobre todo en caliente o en producción.

10.2 RECOMENDACIONES

1. El personal que estará a cargo del uso y administración de los controles de versionamientos y de cambios sugieran mejoras en el incremento de nuevas funcionalidades necesarias y fundamentales para el desarrollo de la empresa incrementando su rendimiento, que garantice la satisfacción del cliente.

2. El personal involucrado posteriormente puede realizar un manual de mejores prácticas, para compartir con los demás compañeros para seguir retroalimentándose del negocio y de la funcionalidad de cada sistema de software desarrollado o mejorado.

3. Las empresas al optar por este método como valor agregado a la calidad de su servicio debe realizar el esfuerzo necesario para implementar la aplicabilidad de Six Sigma en sus procesos de pruebas , y así verificar en que porcentaje es su incremento hacia un nivel máximo de calidad.

4. Analizar la posibilidad de implementar este nuevo modelo de testing con la metodología de programación extrema para aseguramiento de un adecuado producto dentro de todos los estándares y requerimientos de entrada dadas por el cliente, en los plazos establecidos, para así lograr incrementar la satisfacción del cliente, prestigio de la empresa, ampliar a un mercado internacional, tratando de continuamente estar en mejoramiento continuo.

11 Bibliografía

- http://jummp.wordpress.com/2011/09/06/desarrollo-de-software-testing-modelo-en-v/
- http://www.sistedes.es/TJISBD/Vol-3/No-4/articles/pris-09-calvo-cmmidev.pdf
- http://www.fceia.unr.edu.ar/ingsoft/testing-intro-a.pdf
- http://www.ces.com.uy/documentos/imasd/Tesis-Beatriz_Perez_2006.pdf(principal1)
- http://easydevelopmentfiis.blogspot.com/2009/12/software-development-methodologies-and.html
- http://es.scribd.com/doc/24806034/Documentacion-en-el-proceso-de-testing
- http://www.slideshare.net/aracelij/pruebas-de-software(importante2)
- http://www.buenastareas.com/ensayos/Herramientas-Para-Testing-Verificaci%C3%B3n-y-Validaci%C3%B3n/36692.htmlhttp://www.slideshare.net/odelorenzi/testing-software(importante3)
- http://es.wikipedia.org/wiki/Modelo_en_cascada
- http://es.wikipedia.org/wiki/Modelo_de_prototipos
- http://html.rincondelvago.com/modelos-de-procesos-de-software.html
- http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos
- http://es.scribd.com/doc/24806034/Documentacion-en-el-proceso-de-testing

- http://www.calidadysoftware.com/testing/bpm_business_process_management_bizagi.php
- http://ayudantiasubb.files.wordpress.com/2008/06/requerimientos-no-funcionales-generales.pdf
- http://www.mitecnologico.com/Main/EspecificacionesDeRequerimientos
- http://io.us.es/cio2002/comunicaciones/4-%20Gesti%C3%B3n%20de%20la%20Producci%C3%B3n/C061.pdf
- http://plaza-entretenimiento.blogspot.es/
- http://www.lsi.us.es/~javierj/investigacion_ficheros/NDEAv25.pdf
- http://www.nosolounix.com/2010/02/herramientas-test-software.html
- http://materias.fi.uba.ar/7526/docs/teoria.pdf

12 Anexos

ANEXOS