UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO DE UN PROCESO DE PRUEBAS

TESIS

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN

PRESENTA:
EDUARDO VILLEGAS TRUJILLO

DIRECTORA DE TESIS:
ING. LUCILA PATRICIA ARELLANO MENDOZA

MÉXICO D.F. 2007

Agradecimientos

A mi papá, Gilberto Villegas, quien siempre estuvo presente para apoyarme y animarme a seguir adelante, con mucho cariño le dedico este trabajo.

Con especial agradecimiento a mi mama, María del Refugio†, por su espíritu de sacrificio y ser mi ángel, con gran amor.

A mi hermanos Gilberto, Karina, Hilda y Noemí por su incondicional apoyo.

A Gaby y Andy. Su amor y su presencia han sido mi mayor estimulo de lucha, son lo mejor que me ha pasado en la vida.

A mis amigos, por todos los momentos divertidos y de tensión que vivimos a lo largo de mi carrera.

A mi directora de tesis: Ing. Lucila Patricia Arellano Mendoza, por todo el apoyo brindado, para culminar esta tesis.

A los profesores que me han apoyado con la revisión de mi tesis.

A todas las personas que hicieron realidad este trabajo, no solamente la tesis y la carrera, sino toda mi trayectoria.

Eduardo Villegas

ÍNDICE TEMÁTICO

INTRODUCCIÓN

Actualmente la industria, el comercio, organizaciones de servicio y tecnología están caracterizadas por la competencia empresarial de los bienes y servicios. Los cuales interactúan en un medio ambiente de constantes cambios:

- La evolución de la tecnología y procesos.
- Las expectativas de los clientes.
- La creación e innovación de nuevos productos y servicios.
- La idea conceptual de competencia

La corriente de competitividad de hoy en día tiene su base fundamental en el sentido de la **calidad.**

No siendo la excepción las compañías que soportan su operación en desarrollos de software, inician una etapa de calidad en sus sistemas de punto de venta, considerado como un elemento crítico. Esta percepción en el software es el reflejo inherente de los problemas presentados en el área de informática: plazos y presupuestos incumplidos, insatisfacción del usuario, escasa productividad y la baja calidad en los proyectos. En pocas palabras consiste en tratar de cambiar la idea de un simple proceso no sistemático a un proceso determinado por la planificación y la metodología automatizada.

En la mayoría de las empresas actuales en México, el proceso de pruebas se realiza mientras se está construyendo la aplicación, pocas tienen definido una metodología propia para esta fase, pues requiere de una considerable cantidad de tiempo y esfuerzos.

Teniendo como antecedente esto, esta tesis se sitúa en al ámbito de tecnologías de información, diseñando un proceso de pruebas, en la que se utilizan técnicas y herramientas para detectar niveles inadecuados de calidad, aplicando una cantidad de recursos limitados (en especial tiempo y dinero) de forma tal que genere un valor agregado en el proceso de desarrollo de software.

Objetivo

El presente documento tiene como objetivo proporcionar una panorámica amplia de las pruebas de software; permitiendo a los integrantes de las áreas de Tecnología de Información profundizar en metodologías y prácticas que mejoren la calidad de los productos de software mediante su aplicación.

Para conocer la estructura, diseño, aplicación y la razón de ser de las pruebas del software se tienen que cubrir los siguientes puntos:

- Presentar el proceso efectivo de pruebas de software y las metodologías relacionadas que mostrarán cómo hacer un proceso de pruebas adecuado y exitoso, en los proyectos y mantenimiento de desarrollo de software.
- Identificar el ciclo de vida de las pruebas de software y su relación con el ciclo de vida de desarrollo de sistemas.
- Identificar y describir las actividades, y los diferentes tipos de pruebas de software.
- Describir los roles y responsabilidades de los participantes del proceso de pruebas.
- Conocer el proceso formal y adecuado para la realización de pruebas en un proyecto de software.
- Tener mayor conciencia sobre la necesidad imperante de tener un proceso de pruebas formal y definido, que permita registrar oportunamente los errores existentes garantizando su rápida y efectiva corrección.

Contenido

El contenido por capítulo es el siguiente:

1. **Fundamentos teóricos**. El primer capítulo de la tesis contiene los fundamentos teóricos relacionados a la definición del proceso de pruebas de software, así como su propósito. Se describen las diferentes etapas de pruebas, sus tipos y técnicas utilizadas. Los modelos de pruebas más usados en el mercado.

2. **Situación actual**. En este capítulo se describen algunas de las metodologías de desarrollo de software existentes en el mercado, describiendo de forma muy global cada uno de sus procesos, finalizando en una descripción de sus procesos de aseguramiento de calidad.

3. **Planteamiento del problema**. En el tercer capítulo se hace una recopilación y análisis de información referente a los procesos de pruebas dentro de la industria mexicana, con el fin de identificar las áreas de oportunidad.

4. **Propuesta de solución**. El capítulo cuatro propone un proceso de pruebas que cumpla con las necesidades de las áreas de Tecnología de la Información.

5. **Implantación del proceso de pruebas**. Se describen los procesos generados, así como sus entregables por cada fase y el acoplamiento en la metodología de desarrollo de sistemas.

6. **Instalación y puesta en marcha**. El capítulo seis muestra los programas y propuestas de capacitación para la instalación y puesta en marcha del proceso de pruebas diseñado, así como las ventajas y beneficios que se obtienen con el uso del proceso de pruebas diseñado.

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS

1.1. INTRODUCCIÓN

El objetivo de la fase de pruebas de un programa es el de detectar todo posible mal funcionamiento antes de que entre en producción. Un error detectado en las etapas iniciales puede ser costoso de reparar; pero siempre es peor que el error le aparezca al usuario final en producción.

Teniendo como antecedente esto, como parte de sus funciones las áreas de Tecnología de la Información han venido implementando procesos de mejora continua para lograr productos de calidad. ¿Pero en qué consiste un proceso de calidad en un sistema de software? ¿Cómo se realizan las pruebas de software? ¿Debe desarrollarse un plan formal de pruebas? ¿Debe probarse el programa como un todo o sólo deben aplicarse pruebas a una pequeña parte? Éstas y muchas otras preguntas se responderán en este capítulo, el cual contiene los fundamentos teóricos relacionados a la definición del proceso de pruebas de software, se describen los diferentes etapas de pruebas, sus tipos y técnicas utilizadas. Así como también, los modelos de pruebas más usados en el mercado.

1.2. PRUEBAS DE SOFTWARE

1.2.1. ¿Qué son las pruebas de software?

Las pruebas de software es un proceso que corre en paralelo al proceso de desarrollo de software, y que se realiza por el convencimiento de que todo sistema debe ser *"revisado"* con el objetivo de establecer si el nivel de calidad requerido es alcanzado.

En ese proceso se ejecuta el sistema a probar bajo condiciones específicas y se le aplica un conjunto de estímulos diseñados de manera sistemática, con el objetivo de detectar insatisfacción de los requerimientos planteados; todas las actividades y sus resultados deben ser documentadas.

Este proceso debe llevarse a cabo de una manera sistemática, y debe respaldarse en métricas bien definidas.

Al hablar de calidad nos referimos a:

- El grado en que el producto satisface los requerimientos funcionales y no funcionales explícitamente establecidos.
- El nivel al que se siguieron los estándares de desarrollo explícitos y documentados.
- Que el producto muestre las características implícitas que se espera de todo software desarrollado profesionalmente.

Este proceso de prueba genera información muy valiosa acerca de la madurez del producto de software que, junto con aquella acerca del proceso de desarrollo, proporciona una visión concreta de las características del producto a liberar.

1.2.2. Propósito de las pruebas de software

Las pruebas de software son las actividades más visibles en el Aseguramiento de Calidad (SQA) y consisten en encontrar la mayor cantidad de errores posibles, mientras más errores, mejor; describir el error, localizarlo, buscar sus causas y repararlo. Ese es el valor que añade el proceso de pruebas.

Los principales objetivos que se buscan con la prueba de software suelen ser:

- Conocer el nivel de calidad de productos intermedios, para actuar a tiempo (rehacer un componente); esto facilita una administración realista del *time to market*[1] del producto en cuestión.
- No aceptar un producto de software sino hasta que alcance el nivel de calidad pactado; esto eleva el nivel de certidumbre en los usuarios finales del software, y minimiza riesgos.
- Disminuir la penosa y costosa labor de soporte a usuarios insatisfechos, consecuencia de liberar un producto inmaduro. Esto puede mejorar la imagen de la organización desarrolladora (y la credibilidad en ella).
- Reducir costos de mantenimiento (la fase más costosa del desarrollo de software), mediante el diagnóstico oportuno de los componentes del sistema (seguimiento a estándares, legibilidad del código, integración adecuada de los componentes, rendimiento apropiado, nivel y calidad del reuso, calidad de la documentación, etc.).
- Obtener información concreta acerca de fallas, que pueda usarse como apoyo en la mejora de procesos, y en la de los desarrolladores (capacitación en áreas de oportunidad).

Entre más pronto se apliquen mecanismos de prueba en el proceso de desarrollo, más fácilmente podrá evitarse que el proyecto se salga del tiempo y presupuesto planeado, pues se podrán detectar más problemas originados en las fases tempranas del proceso, que son los que mayor impacto tienen.

¿Por qué probar el software?

El nivel de esfuerzo de las pruebas, depende del riesgo involucrado en la operación de los sistemas, así como también de los siguientes aspectos:

- Calidad del software.
- Cambios de la tecnología.
- Intentar obtener software libre de errores.
- Competitividad comercial.
- Reducción de costos y riesgos.
- Incremento de la productividad.

Algunas razones:

- Tener mayor calidad aumenta la fidelidad de nuestros clientes
- Siempre es más barato mantener un cliente que conseguir uno nuevo.
- ¿Cuánto perdemos cuando perdemos un cliente?

[1] **Time to market**. Es el tiempo que transcurre entre que un producto es concebido y está disponible para la venta.

¿Por qué es importante?

Con frecuencia, las pruebas requieren una mayor cantidad del esfuerzo dedicado al proyecto que cualquier otra actividad de ingeniería de software. Si se realiza sin un plan, se desperdicia tiempo, se dedica un esfuerzo innecesario y, aún peor, es posible que no se detecten errores. Por tanto, lo razonable sería establecer una estrategia sistemática para probar el software.

¿Cuáles son los pasos?

La prueba empieza por lo “pequeño” y avanza hacia lo “grande”. Esto significa que, en las primeras etapas, la prueba se concentra en un solo componente o en un grupo pequeño de componentes relacionados y se aplica para descubrir errores en la lógica de datos y del procesamiento que se ha encapsulado en el componente. Una vez probados los componentes, deben integrarse hasta que todo el sistema se haya construido. En este punto se ejecuta una serie de pruebas de alto nivel para descubrir errores en la satisfacción de los requisitos del cliente. A medida que se descubren, los errores deben diagnosticarse y corregirse empleando un proceso llamado depuración.

1.2.3. Etapas de pruebas

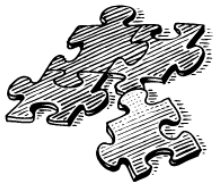
Pruebas unitarias

Las pruebas unitarias tienen como objetivo comprobar que cada uno de los módulos funciona de acuerdo a las especificaciones dadas por el cliente.

Para la etapa de pruebas integración, se debe considerar que son pruebas aisladas de un programa específico. Se debe elaborar la documentación del proceso de revisión de sintaxis y validación de las modificaciones.

La información debe contemplar la descripción de las entradas (parámetros, archivos, bases de datos, etc.), las salidas (archivos, bases de datos, reportes, etc.) y las especificaciones precisas en cuanto a las correcciones.

Pruebas de integración



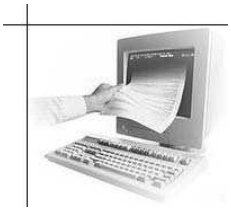
Se realizan al unir los diferentes componentes del sistema y revisar su comportamiento, así como su buen funcionamiento.

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

Tipos de pruebas de integración:

- *Integración ascendente*: Empieza del último módulo al módulo del mando principal
- *Integración descendente*: Empieza con el módulo del mando principal hacia el último módulo
- *Integración tipo sándwich*: Es la integración combinada entre la ascendente y la descendente

Pruebas de aceptación



Estas pruebas las realiza el cliente y/o usuario. 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, sino una vez pasadas 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. Los desarrolladores se suelen llevar las manos a la cabeza:

"Pero, ¿a quién se le ocurre usar así mi programa?"

Sea como sea, el cliente siempre tiene razón. Decir que los requisitos no estaban claros, o que el manual es ambiguo puede salvar la cara; pero ciertamente no deja satisfecho al cliente. Alegar que el cliente es un inútil es otra tentación muy fuerte, que conviene reprimir. Por estas razones, 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.

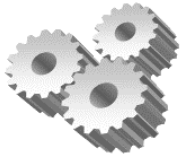
Las pruebas beta vienen después de las pruebas alfa, y se desarrollan en el entorno del cliente, un entorno que está fuera de control. Aquí el cliente se queda a solas con el producto y trata de encontrarle fallos (reales o imaginarios) de los que informa al desarrollador.

Las pruebas alfa y beta son habituales en productos que se van a vender a muchos clientes. Algunos de los potenciales compradores se prestan a estas pruebas bien por ir entrenando a su personal con tiempo, bien a cambio de alguna ventaja económica (mejor precio sobre el producto final, derecho a mantenimiento gratuito, a nuevas versiones, etc.). La experiencia muestra que estas prácticas son muy eficaces.

La carta que firma el usuario es muy importante ya que queda por escrito que están satisfechos y que no encontraron ningún error. Se hacen tres copias de esta carta; una para el usuario, otra para la carpeta del cliente y otra para la carpeta del proyecto.

Les permiten a los clientes y usuarios del sistema determinar si éste realmente satisface sus necesidades y expectativas. Las pruebas de aceptación son escritas, conducidas y evaluadas por el cliente.

Pruebas de regresión



Pruebas para comprobar que las modificaciones no generaron errores donde antes no los había.

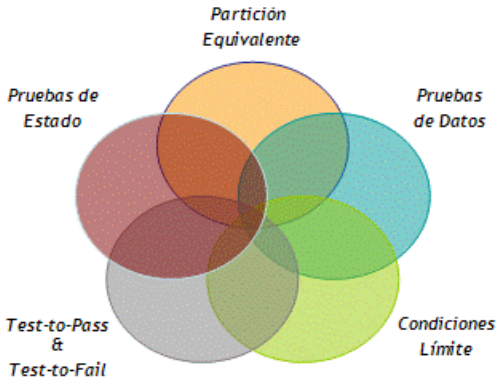
1.2.4. Técnicas de pruebas

Pruebas de caja negra

Realizadas directamente sobre la interfaz de usuario manipulando los datos de entrada y revisando los resultados. Están basadas en los requerimientos y la funcionalidad, no se necesitan conocimientos acerca del diseño interno o del código.

Estas pruebas se centran principalmente en los requisitos funcionales del software. Por lo tanto, la prueba de caja negra permite al ingeniero de pruebas obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa

Siendo esta técnica la más utilizada por los profesionales de pruebas, es importante mencionar que a su vez tiene diferentes subtécnicas.



Test to pass: Asegurar que los procesos fundamentales (mínimos) de software funcionan correctamente. (Happy path[2])

Test to fail: Diseñar y ejecutar pruebas con el propósito de corromper el software.

Partición equivalente: Conjunto de condiciones de pruebas que prueban lo mismo o revelan el mismo error.

Características:

- Agrupar las entradas, salidas (resultados) y procesos del software que sean similares.
- Si se realiza esta partición demasiado tarde se corre el riesgo de eliminar pruebas que podrían detectar defectos.

Pruebas de datos: Probar que las entradas de usuario, resultados que recibe y cualquier resultado intermedio al software es manejado correctamente. Por ejemplo: entradas desde el teclado, clic del mouse, archivos de disco, impresión, etc.

[2] **Happy path**. Es cuando solo se prueba el recorrido del flujo básico en una aplicación.

Condiciones límite: Situaciones en el borde de los límites operacionales previstos del software.

- Tipos: Numéricos, caracteres, posición, cantidad, velocidad, localización y tamaño
- Características: Primero/último, vacío/lleno, lento/rápido, grande/pequeño, mínimo/máximo, alto/bajo, etc.

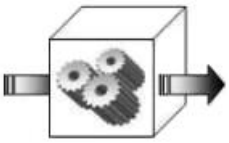
Ejemplo: si un campo de entrada (text box) permite de 1 a 255 caracteres, intentar incorporar un carácter y 255 caracteres como condición valida. También intentar 254 caracteres como elección válida. Ingresar 0 y 256 caracteres como las condiciones inválidas.

Mapa de transición de estados: Representación gráfica de un estado, condiciones, procesos y estados adyacentes.

Características:

- Se debe de mostrar el estado en el que el software puede estar.
- Entrada o condición para pasar a otro estado.
- Condiciones o resultados al entrar o salir de un estado.
- Estos mapas puedes ser creados con diferentes técnicas.

Pruebas de caja blanca



El Ingeniero de pruebas se cerciorará del proceso que el sistema seguirá para conseguir los resultados esperados. En los métodos de caja blanca determinamos cuáles son los casos de prueba a partir del código fuente del software y se utilizan las especificaciones para determinar el resultado esperado del caso.

En estas pruebas estamos siempre observando el código, que las pruebas se dedican a ejecutar con ánimo de "probarlo todo". Esta noción de prueba total se formaliza en lo que se llama "cobertura" y no es sino una medida porcentual de ¿cuánto código hemos cubierto?

Pruebas estáticas

Infraestructura técnica (hardware, código, red), organización (Centro de Cómputo, procedimientos de seguridad, etc.), documentación (manuales, descripción, procedimientos, etc.), Implementación (esquemas de operación, capacitación, soporte, etc.).

Pruebas dinámicas

Pruebas realizadas con el programa en ejecución tal y como lo utilizaría el usuario, ejecutando casos de prueba y matrices.

1.2.5. Tipos de pruebas

Pruebas funcionales

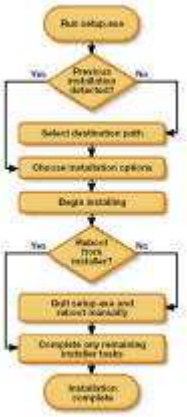
Asegurar que el sistema integrado ejecute sus funciones de acuerdo a lo establecido por las especificaciones de requerimientos funcionales.

Entiéndase requerimientos funcionales como **todo** lo que el cliente pidió que realizara el sistema, y que se presenta de manera explícita en el documento.

¿Se ha construido el producto correcto?

Tipo de prueba más importante y más aplicada en la disciplina de pruebas de software.

Pruebas de instalación / desinstalación

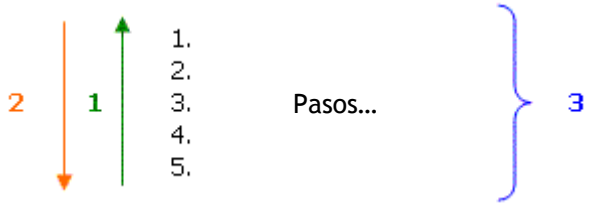


Este tipo de prueba tiene como objetivo el asegurar el correcto funcionamiento del instalador de la aplicación. Verificar que se instalen completamente cada uno de los componentes y que de la misma forma se realice la desinstalación.

Dentro de las pruebas de instalación se recomienda probar los siguientes escenarios:

- PC "limpia".
- Una nueva versión sobre una antigua.
- Una nueva versión sobre otra nueva versión.
- Reinstalación.
- Cambiar la trayectoria de instalación.

Para el caso específico de la desinstalación se recomienda remover los componentes que contenga la aplicación en diferente orden (**ascendente**, **descendente** y **aleatorio**) y verificar que esto no afecte al sistema operativo o a futuras instalaciones de la aplicación.



Pruebas de configuración

Tipo de pruebas centrado a verificar el correcto funcionamiento de la aplicación en diferente hardware y software.

Pruebas de seguridad

Pruebas que verifican principalmente que los datos (o sistemas) son accesibles solamente por los actores previstos.

También tiene que ver con verificar el grado de vulnerabilidad que tiene un sistema.

La seguridad puede ser probada a diferentes niveles:

- Redes.
- Seguridad de la aplicación del lado del cliente.
- Seguridad de la aplicación del lado del servidor.

Pruebas de datos (integridad de la información)

Comprobar que la información se administra, guarda y manipula de manera correcta dentro de la aplicación y/o base de datos.

Pruebas de perfomance

Es importante verificar el tiempo de respuesta, u otros parámetros de gasto de las aplicaciones. Típicamente nos puede preocupar:

¿Cuánto tiempo le lleva al sistema procesar tantos datos?, ¿Cuánta memoria consume?, ¿Cuánto espacio en disco utiliza?, ¿Cuántos datos transfiere por un canal de comunicaciones? Para todos estos parámetros suele ser importante conocer cómo evolucionan al variar la dimensión del problema (por ejemplo, al duplicarse el volumen de datos de entrada).

Estas pruebas se hacen una vez que se ha probado que el sistema funciona correctamente en condiciones ideales, ya que las pruebas de performance son para probarlo en condiciones extremas o poco comunes.

Las pruebas de performance se dividen principalmente en:

- **Pruebas de stress:** Son pruebas que consisten repetir innumerables veces una acción o simular el uso excesivo de la aplicación.
- **Pruebas de volumen:** Estas pruebas se realizan con una gran carga de datos para determinar en que punto el tiempo de respuesta del software aumenta o falla totalmente.

Pruebas de usabilidad

Evaluar la utilidad y robustez de herramientas de software e interfaces sistemáticamente por un rango de plataformas de escritorio comunes.

Se debe de probar que la aplicación sea lo suficientemente amigable con el usuario final, para llevar a cabo estas pruebas se debe saber hacia que tipo de usuarios esta diseñado el software desarrollado.

La usabilidad de un sistema es cuestión de saber mejorar un sistema basado en las necesidades y expectativas de los usuarios.

Otros tipos de pruebas

Pruebas de recorridos (walkthroughs). Quizás es una técnica más aplicada en control de calidad que en pruebas. Consiste en sentar alrededor de una mesa a los desarrolladores y a una serie de críticos, bajo las órdenes de un moderador. El método consiste en que los revisores lean el programa línea a línea y pidan explicaciones de todo lo que no esté claro. Puede que simplemente falte un comentario explicativo, que detecten un error auténtico o que simplemente el código sea tan complejo de entender/explicar que más vale que se rehaga de forma más simple. Para un sistema complejo pueden hacer falta muchas sesiones.

Esta técnica es muy eficaz localizando errores de naturaleza local; pero falla estrepitosamente cuando el error deriva de la interacción entre dos partes alejadas del programa.

Pruebas aleatorias (random testing). Comenzar la fase de pruebas con una serie de casos elegidos al azar. Esto pondrá de manifiesto los errores más patentes. No obstante, pueden permanecer ocultos errores que sólo se muestran ante entradas muy precisas.

Si el programa es poco crítico (una aplicación personal, un juego) puede que esto sea suficiente. Pero si se trata de una aplicación corporativa, de misión crítica, militar o con riesgo para vidas humanas, definitivamente seria insuficiente.

Pruebas de solidez (robustness testing). Se prueba la capacidad del sistema para salir de situaciones embarazosas provocadas por errores en el suministro de datos. Estas pruebas son importantes en sistemas con una interfaz al exterior, en particular cuando la interfaz es humana.

Por ejemplo, en un sistema que admite una serie de órdenes se deben probar los siguientes extremos:

- Ordenes correctas, todas y cada una.
- Ordenes con defectos de sintaxis, tanto pequeñas desviaciones como errores de bulto.
- Ordenes correctas, pero en orden incorrecto, o fuera de lugar.
- La orden nula (línea vacía, una o más).
- Ordenes correctas, pero con datos de más.
- Provocar una interrupción (BREAK, ^C, o lo que corresponda al sistema soporte) justo después de introducir una orden.
- Ordenes con delimitadores inapropiados (comas, puntos).
- Ordenes con delimitadores incongruentes consigo mismos (por ejemplo, esto]).

Pruebas de conformidad u homologación (conformance testing). En programas de comunicaciones es muy frecuente que, además de los requisitos específicos del programa que estamos construyendo, aparezca alguna norma más amplia a la que el programa deba atenerse. Es frecuente que organismos internacionales como ISO (International Standards Organization) y el CCITT (International Consultative Committee of Telegraph and Telephone) elaboren especificaciones de referencia a las que los diversos fabricantes deben atenerse para que sus ordenadores sean capaces de entenderse entre sí.

Estas pruebas suelen realizarse en un centro especialmente acreditado al efecto y, si se pasan satisfactoriamente, el producto recibe un sello oficial y/o certificación.

Interoperabilidad (interoperability testing). En el mismo escenario del punto anterior, programas de comunicaciones que deben permitir que dos computadoras se entiendan, además de las pruebas de conformidad se suelen correr una serie de pruebas, también utilizando la técnica de caja negra, que involucran dos o más productos, y buscan problemas de comunicación entre ellos.

Mutación (mutation testing). Es una técnica curiosa consistente en alterar ligeramente el sistema bajo pruebas (introduciendo errores) para averiguar si nuestra batería de pruebas es capaz de detectarlo. Si no, más vale introducir nuevas pruebas. Todo esto es muy laborioso y francamente artesano.

1.2.6. Etapas, técnicas y tipos de pruebas

Una parte fundamental de las pruebas de software (proceso) es delimitar los tipos de pruebas que se van a aplicar a un sistema, y para esto, es necesario primero entender como están delimitadas.

Muchos autores hablan únicamente de tipos de pruebas, pero después de un análisis, por lo que a continuación se muestra una división más elaborada que ayuda a delimitar de mejor manera el momento (etapa) en donde utilizando diferentes técnicas (¿Cómo?) aplicamos diferentes tipos de pruebas (¿Qué?), de esta manera nos aseguramos que una aplicación funcione correctamente. Es por eso que hablamos de etapas, técnicas y tipos de pruebas, las cuales están divididas de la siguiente manera:

Etapa	Técnica de prueba	Tipos de pruebas						
		Funcionalidad				Performance		Usabilidad
		Configuración	Funcionales	Instalación / Desinstalación	Integridad de datos	Estrés	Volumen	Usabilidad
Pruebas unitarias	Pruebas de caja negra	✓	✓	✓	✓	✓	✓	×
Pruebas unitarias	Pruebas de caja blanca	✓	✓	✓	✓	✓	✓	×
Pruebas unitarias	Pruebas estáticas	✓	✓	×	✓	×	×	✓
Pruebas de integración	Pruebas de caja blanca	✓	✓	✓	✓	✓	✓	×
Pruebas de integración	Pruebas estáticas	✓	✓	×	✓	×	×	✓
Pruebas de regresión	Pruebas estáticas	✓	✓	×	✓	×	×	✓
Pruebas de regresión	Pruebas dinámicas	✓	✓	✓	✓	✓	✓	✓
Pruebas de aceptación	Pruebas dinámicas	✓	✓	✓	✓	✓	✓	✓
Pruebas de aceptación	Pruebas aleatorias	✓	✓	✓	✓	✓	✓	✓

Etapas	• Pruebas unitarias • Pruebas de integración • Pruebas de regresión • Pruebas de aceptación
Técnicas	• Pruebas de caja negra • Pruebas de caja blanca • Pruebas estáticas • Pruebas dinámicas

Tipos	• Pruebas de requerimientos
	• Pruebas de confiabilidad
	  ▪ Pruebas de integridad
	  ▪ Pruebas de estructura
	• Pruebas de funcionalidad:
	  ▪ Pruebas de configuración
	  ▪ Pruebas funcionales
	  ▪ Pruebas de instalación/desinstalación
	  ▪ Pruebas de seguridad
	• Pruebas de performance (comparación, contención, carga, stress y volumen)
	• Pruebas de usabilidad (pruebas de facilidad de uso)

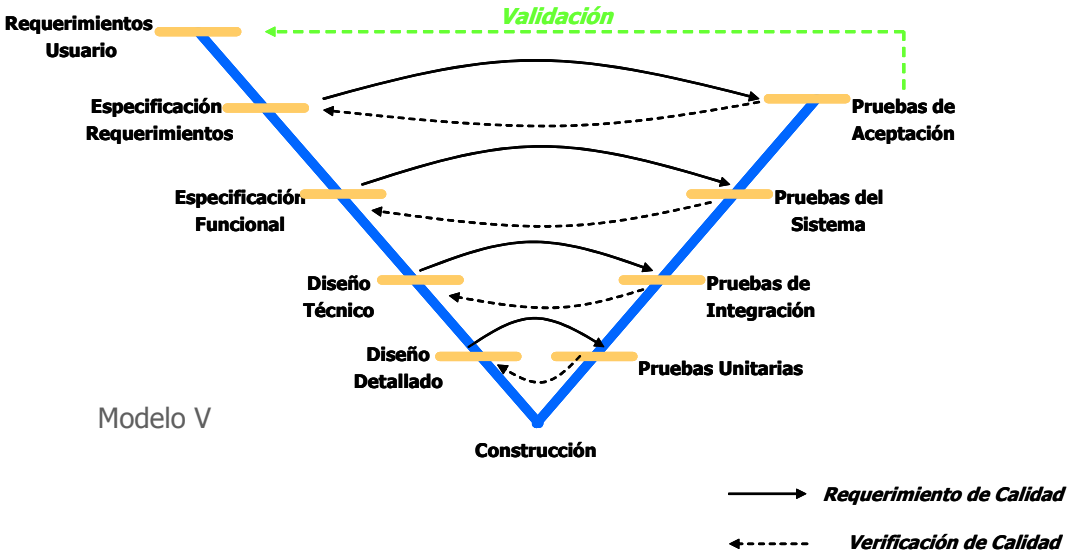
1.3. MODELOS DE PRUEBAS

1.3.1. Modelo V

El modelo V fue desarrollado para regular el proceso de desarrollo de software por la Administración Federal Alemana. Describe las actividades y los resultados que se producen durante el desarrollo del software.

El modelo V es una representación gráfica del ciclo de vida del desarrollo del sistema. Resume los pasos principales que hay que tomar en conjunción con las correspondientes entregas de los sistemas de validación.

La parte izquierda de la **V** representa la corriente donde se definen las especificaciones del sistema. La parte derecha de la **V** representa la corriente donde se comprueba el sistema (contra las especificaciones definidas en la parte izquierda). La parte de abajo, donde se encuentran ambas partes, representa la corriente de desarrollo.



La corriente de especificación consiste principalmente de:

- Especificaciones de requerimiento de usuario
- Especificaciones funcionales
- Especificaciones de diseño

La corriente de pruebas, por su parte, suele consistir de:

- Calificación de instalación
- Calificación operacional
- Calificación de rendimiento

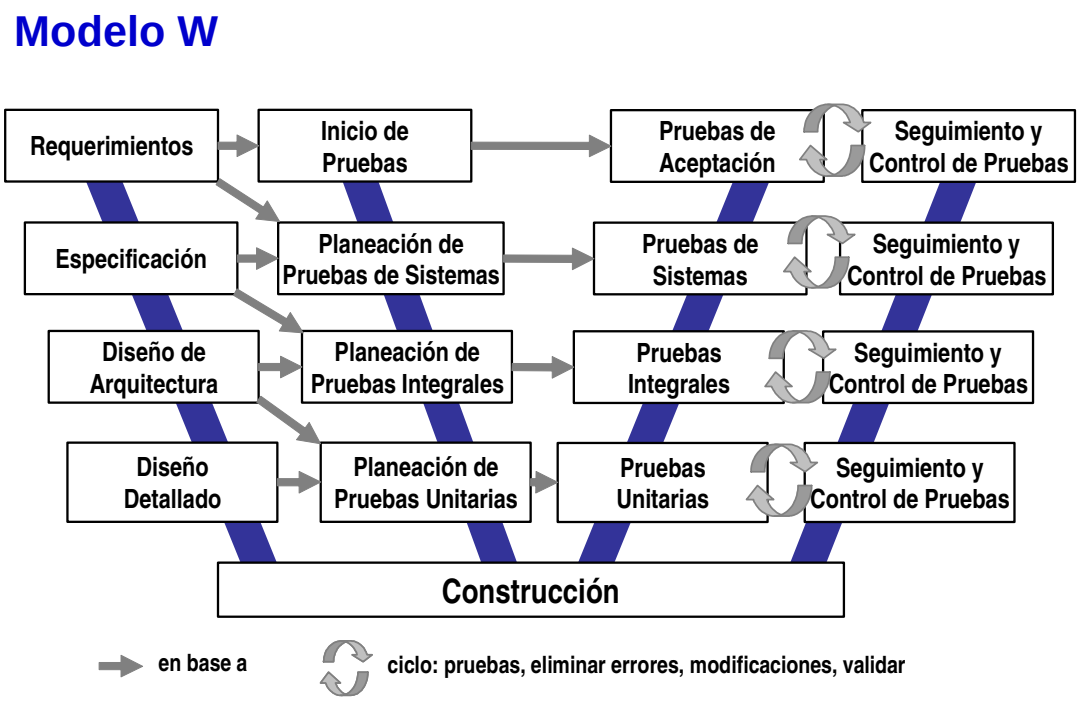
La corriente de desarrollo puede consistir (depende del tipo de sistema y del alcance del desarrollo) en personalización, configuración o codificación.

1.3.2. Modelo W

El **modelo W**, es un proceso de pruebas que se ejecuta en paralelo al proceso de desarrollo.

Basándose en el modelo V, esta sección describe cómo se relacionan las tareas de pruebas con las tareas de desarrollo. Este modelo de pruebas, denominado W, clarifica también la prioridad de las tareas y la dependencia entre las actividades de desarrollo y las de pruebas. Aun siendo tan simple como el modelo V, el W hace patente la importancia de ordenar de las actividades individuales de pruebas. Asimismo, evidencia que pruebas no es equivalente a eliminación de errores.

A continuación se muestra el diagrama de contexto de modelo W.

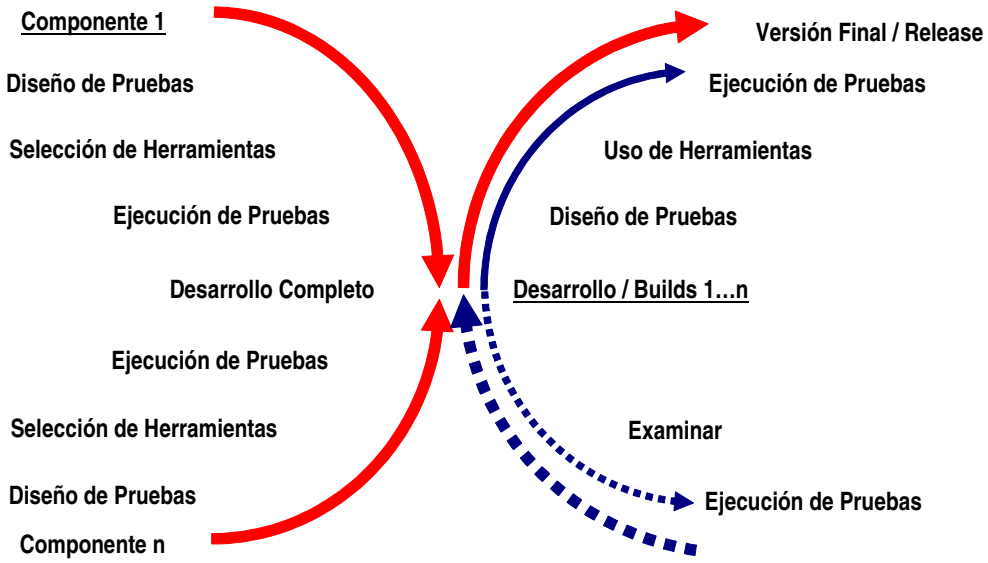


1.3.3. Modelo X

El modelo X entra en la clasificación de los “nuevos modelos de pruebas”, en la actualidad existe muy poca literatura acerca de este modelo. El modelo X pretende llenar los huecos de las actividades de la izquierda abiertos en el modelo V.

El lado izquierdo del modelo X representa la codificación y las prueba separadas de los componentes individuales, continuando con la integración de las componentes en estructuras (cuadrante superior derecho) que también deben ser probadas. Las estructuras que pasan sus pruebas integrales se pueden pasar a los usuarios o para la inclusión en estructuras más grandes para continuar con pruebas de integración con mayor cobertura, hasta llegar a las pruebas de aceptación del usuario final. Las líneas se curvan para indicar que las iteraciones pueden ocurrir con respecto a todas las piezas.

A continuación se muestra el diagrama de contexto de modelo X.



Según lo considerado en el gráfico anterior, el modelo X también trata la prueba exploratoria (cuadrante inferior derecho). Aquí se plantea "¿qué pasa si pruebo esto?", este tipo de acciones permiten a menudo a probadores experimentados encontrar más errores yendo más allá de las pruebas planeadas.

Sin embargo, el centrarse en tales actividades también crea preocupaciones. Un modelo no es igual que un plan del proyecto individual. Un modelo no puede, y no se debe esperar, que describa detalladamente cada tarea en cada proyecto.

El modelo X incluye los pasos de diseño de pruebas, así como las actividades relacionadas con usar los varios tipos de herramientas de prueba, mientras que el modelo V no lo contempla.

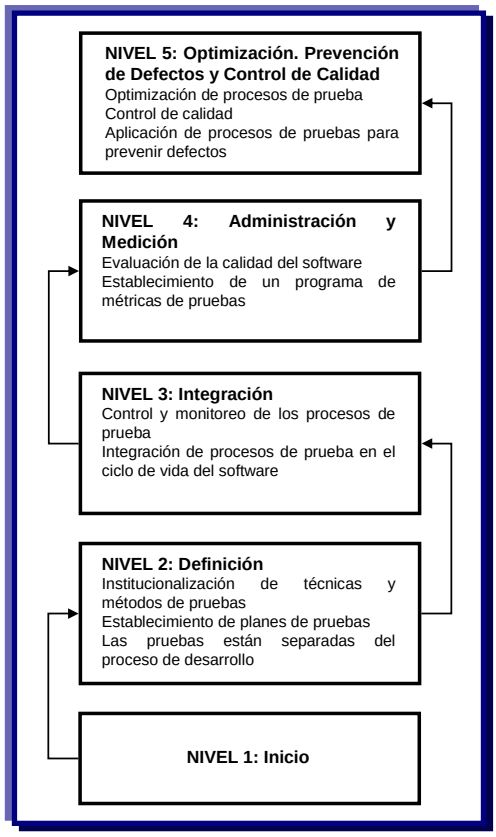
1.3.4. Modelo madurez de pruebas (TMM)

Algunas características:

- Modelo de madurez (TMM: Testing Maturity Model) que ayuda y guía a las organizaciones a focalizarse en la evaluación y mejora de su staff y proceso de pruebas.
- TMM es la contraparte de CMM pero específicamente para pruebas de software.
- Desarrollado en el Instituto de Ingeniería de Illinois

NIVELES DE MADUREZ

A continuación se listan los niveles de madurez del modelo de madurez de pruebas:



Nivel 1. Inicio
En este nivel las pruebas se consideran como parte del desarrollo y no existe un proceso formal. El objetivo de las pruebas en este nivel es demostrar que las aplicaciones funcionan sin fallas graves. Hay una falta de recursos, herramientas y testers entrenados.

Nivel 2. Definición
Hay un proceso definido y las pruebas están claramente separadas del proceso de desarrollo. Se establecen planes y estrategias de pruebas. Se aplican técnicas formales para el diseño de casos de prueba derivados de los requerimientos. En este nivel, las pruebas todavía se inician tarde en el ciclo de vida de un proyecto.

Nivel 3. Integración
Las pruebas están totalmente integradas a ciclo de vida de un proyecto. La planeación de pruebas se elabora en etapas tempranas y se genera una especificación de pruebas. Existen estrategias de pruebas basadas en administración de riesgos y documentación de requerimientos. Existe un área formal así como programas de entrenamiento.

Nivel 4. Administración y Medición
El proceso de pruebas está bien definido, es robusto y se puede medir. Los productos de software son evaluados usando criterios de calidad como confiabilidad, usabilidad y mantenimiento. Los casos de prueba se concentran, almacenan y administran de forma centralizada para su reuso y aplicación en pruebas de regresión.

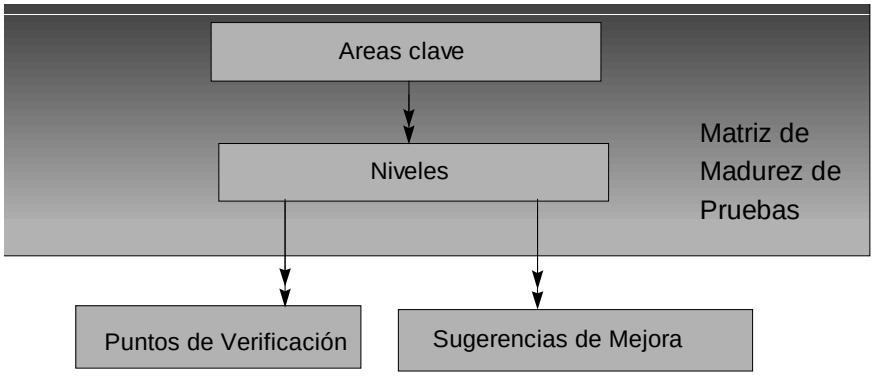
Nivel 5. Optimización
Los métodos y técnicas se van optimizando y hay un continuo esfuerzo en la mejora de los procesos de prueba. Se incluyen procesos como “prevención de defectos” y “control de calidad”. Existen procedimientos para la evaluación y selección de herramientas. Existen herramientas que soportan el proceso de pruebas durante las fases de diseño, ejecución, pruebas de regresión, administración. El proceso de pruebas tiene como objetivo prevenir defectos.

1.3.5. Modelo de mejora del proceso de pruebas (TPI)

El modelo de mejora del proceso de pruebas (TPI - Test Process Improvement) se ha desarrollado partiendo del conocimiento y la experiencia de las pruebas de control de software. El modelo TPI es un medio de ayuda para mejorar el proceso de pruebas. El modelo permite visualizar el nivel de "madurez" del proceso de pruebas dentro de su organización. Partiendo de este criterio, el modelo ayuda a definir pasos de mejora graduales y controlados.

DESCRIPCIÓN DEL MODELO

El modelo es visualizado de la siguiente manera:



Áreas Clave

En cada proceso de pruebas hay ciertas áreas que necesitan atención específica con el fin de lograr un proceso bien definido. Estas **áreas clave** constituyen la base para mejorar y estructurar el proceso de pruebas. El modelo TPI tiene 20 áreas clave. El alcance de la mejora del proceso de pruebas incluye normalmente las pruebas de caja negra, como son las pruebas de sistema y pruebas de aceptación. La mayoría de las áreas clave están dirigidas a ello. Sin embargo, para mejorar procesos de pruebas más "maduros", se debe prestar atención a las actividades de verificación y pruebas de caja blanca, como son las pruebas unitarias y las de integración. Estas pruebas se incluyen como áreas clave separadas con el fin de prestar la atención adecuada a estos procesos.

A continuación se encuentra una lista completa de áreas clave, seguida de una explicación.

- Estrategia de pruebas
- Modelo del ciclo de vida
- Momento de involucrarse
- Estimación y planeamiento
- Técnicas de diseño de pruebas
- Técnicas de pruebas estáticas
- Métricas
- Herramientas de prueba
- Entorno de pruebas
- Entorno de oficina
- Compromiso y motivación
- Funciones de pruebas y capacitación
- Alcance de la metodología
- Comunicación
- Informes
- Manejo de defectos
- Administración de los elementos de prueba (testware)
- Administración del proceso de pruebas
- Revisión estructurada
- Pruebas de caja blanca

Área Clave	Descripción
Estrategia de pruebas	La estrategia de pruebas debe estar dirigida a encontrar los defectos más importantes con la mayor rapidez y el menor costo posible. En la estrategia de pruebas se determinan los defectos descubiertos al realizar las pruebas. Al involucrar más pruebas y medidas de detección, se puede definir mejor la estrategia. Las duplicidades u omisiones involuntarias que tengan lugar entre diferentes pruebas pueden ser evitadas coordinando a los probadores y las actividades de pruebas, así como fijando el alcance de la prueba.
Modelo del ciclo de vida	Dentro del proceso de pruebas se distinguen algunas fases: planeamiento, preparación, diseño, ejecución y terminación. En cada fase se efectúan algunas actividades. Para cada actividad se registran aspectos tales como: objetivo, entradas, proceso, conceptos, dependencias, técnicas y herramientas, instalaciones y documentación. La importancia de contar con un modelo de ciclo de vida reside en tener un mejor control del proceso de pruebas, dado que las actividades pueden ser planeadas y controladas consistentemente.
Momento de involucrarse	Aunque el momento de ejecución de la pruebas se inicia normalmente al concluir el diseño del software, el proceso de pruebas debe iniciarse mucho antes. La involucración temprana de las pruebas en el ciclo de vida del desarrollo de sistemas ayuda a detectar los defectos con mayor antelación y/o con más facilidad, e incluso ayuda a prevenir defectos. Ello redunda en una mejor coordinación entre las pruebas además de reducir enormemente la ruta crítica de las pruebas.
Estimación y planeamiento	Se necesita estimar y planear para definir las actividades que deben realizarse en cada momento y los recursos que serán necesarios. Estimar y planear son la base para ahorrar capacidad y para coordinar las actividades de prueba y las actividades del proyecto.
Técnicas de diseño de pruebas	Una técnica de diseño de pruebas se define como “un modelo para derivar casos de prueba de la documentación”. El uso de estas técnicas incrementa la visibilidad de la calidad y la cobertura de pruebas y conduce a una mayor reusabilidad de pruebas. En base a una estrategia de pruebas, se utilizan diferentes técnicas de diseño de pruebas para obtener la cobertura del código de las partes de software cuyo alcance fue acordado.
Técnicas de pruebas estáticas	No todo puede ni requiere ser probado dinámicamente, por ejemplo, ejecutando los programas. Al fenómeno de validar productos sin ejecutar los programas o evaluar métricas de calidad específicas, se le llama “pruebas estáticas”. Las listas de verificación y mecanismos similares son muy útiles en este caso.
Métricas	Las métricas son observaciones cuantitativas (mediciones). Para el proceso de pruebas, medir el progreso y la calidad del software probado es muy importante, así como también lo son las métricas en estas áreas. Las métricas se utilizan para poder administrar el proceso de pruebas, para poder tener evidencia al momento de expresar una opinión, y también para comparar diferentes sistemas o procesos. Ayudan a contestar preguntas tales como: “¿Por qué el sistema A tiene mucho menos fallas en producción que el sistema B?”; “¿Por qué el proceso de pruebas del sistema A puede ser ejecutado con mayor rapidez y con mayor alcance que el proceso del sistema B?”. En la mejora del proceso de pruebas, las métricas son esencialmente importantes para juzgar los resultados de ciertas acciones de mejora. Esto se hace midiendo antes y después de que la mejora sea implantada.
Herramientas de prueba	La automatización del proceso de pruebas puede efectuarse de muy diversos modos. Como regla, la automatización sirve a alguna de las metas siguientes: - Menor consumo de recursos - Menos consumo de tiempo - Mejor cobertura de pruebas - Mayor flexibilidad - Mayor o mas rápida comprensión del estado del proceso de pruebas - Mayor motivación del personal de pruebas

Área Clave	Descripción
Entorno de pruebas	La ejecución de la prueba tiene lugar en el llamado “entorno de pruebas”. Este entorno de pruebas consta de los siguientes componentes: • Hardware • Software • Instalaciones de comunicaciones • Herramientas para la creación y el uso de datos de prueba • Procedimientos El entorno debe quedar establecido para poder hacer pruebas de forma óptima. El entorno de pruebas influye en gran medida en la calidad, duración y costo del proceso de pruebas. Algunos aspectos importantes del entorno de pruebas son: roles y responsabilidades, control, disponibilidad suficiente y a tiempo, flexibilidad y representatividad de los entornos reales de producción.
Entorno de oficina	El personal de pruebas necesita oficinas, escritorios, sillas, ordenadores, grabadoras, impresoras, teléfonos, etc. Un arreglo adecuado y a tiempo del entorno de oficina influye positivamente en la motivación de dicho personal, y en la comunicación y eficiencia de la ejecución de las tareas de pruebas.
Compromiso y motivación	El compromiso y la motivación del personal involucrado en las pruebas son condiciones indispensables para lograr un proceso de pruebas “maduro”. El personal involucrado en el proceso de pruebas incluye no solamente a los miembros del equipo de pruebas, sino también, entre otros, a líderes de proyectos y a los directivos. El proceso de pruebas debe disponer del tiempo, dinero y recursos suficientes (cuantitativa y cualitativamente) para efectuar una buena prueba. La cooperación y comunicación con los otros miembros del proyecto da como resultado un proceso eficiente y involucrarse temprano.
Funciones de pruebas y capacitación	El personal de pruebas requiere de una cierta preparación. Se requiere una combinación de diferentes materias, funciones, conocimientos y habilidades. Por ejemplo, aparte de tener experiencia específica sobre pruebas, también se requiere conocimiento del sistema que está siendo probado, conocimiento de la organización y conocimiento general de automatización. También es importante contar con ciertas habilidades sociales. Para poder tener estas cualidades, es necesario ofrecer educación y capacitación.
Alcance de la metodología	Para cada proceso de pruebas se utiliza cierta metodología o acercamiento, que consiste en actividades, procedimientos, estándares, técnicas, etc. Si estas metodologías difieren para cada proceso de pruebas en la organización, o si la metodología usada es demasiado genérica, muchas cosas tienen que ser reinventadas una y otra vez. El objetivo de una organización es utilizar una metodología que sea lo suficientemente genérica como para ser ampliamente aplicable, pero al mismo tiempo que sea lo suficientemente detallada para poder evitar las reinvenciones en cada nuevo proceso de pruebas.
Comunicación	En un proceso de pruebas, la comunicación se lleva a cabo de diversos modos, tanto entre los controladores como grupo, como entre los controladores y otros miembros del proyecto, tales como el desarrollador, el usuario final, y el líder del proyecto. Algunos temas objeto de comunicación son la estrategia de pruebas, así como el progreso y la calidad del software bajo prueba.
Informes	Realizar pruebas no está relacionado solamente con la detección de defectos, sino también con informar sobre la calidad (o falta de calidad) del software. Los informes deben dirigirse a proporcionar información bien fundada hacia el proyecto y el cliente sobre la calidad del software e incluso sobre el proceso de desarrollo del software.
Manejo de defectos	Aunque el manejo de defectos es de hecho responsabilidad del líder de proyecto, los probadores están altamente involucrados en ello. Una buena administración debería ser capaz de controlar el ciclo de vida de un defecto y crear diferentes informes (estadísticos). Estos informes se usan para prestar asesoramiento bien fundado sobre la calidad del software.

Área Clave	Descripción
Administración de los elementos de prueba (testware)	El testware o los elementos de prueba deberían ser mantenibles y reutilizables, y por lo tanto, deben ser administrados. Aparte de los elementos de pruebas, también los productos de fases previas, tales como el diseño y la construcción, deben estar bien administrados (¡aunque no por los probadores!). El proceso de pruebas puede verse seriamente interrumpido por la entrega de versiones erróneas de programas, etc. Una buena administración de estos productos incrementa la factibilidad de pruebas (y por ende la calidad) del software.
Administración del proceso de pruebas	Con el fin de administrar cada proceso y cada actividad, son esenciales los cuatro pasos del llamado círculo de calidad de Deming: planear, hacer, comprobar y actuar. Un proceso de pruebas bien administrado es de la mayor importancia para efectuar la mejor prueba posible en la a veces turbulenta área de pruebas.
Revisión estructurada	Revisión Estructurada en este contexto significa validar algunos conceptos tales como el diseño funcional. En comparación con las pruebas, la ventaja de la revisión estructurada es que ofrece la oportunidad de detectar defectos con prontitud. Esto lleva a que los costos de reparación sean considerablemente más bajos. Además, realizar dicha revisión es relativamente simple dado que ningún programa debe ser ejecutado y no se requiere establecer ningún entorno de pruebas, etc.
Pruebas de caja blanca	Una prueba de caja blanca está definida como una prueba de las propiedades internas de un objeto, mediante el conocimiento de funciones internas. Estas pruebas son ejecutadas por los desarrolladores. La prueba unitaria y la de integración son pruebas bastante conocidas de caja blanca. Al igual que la revisión estructurada, estas pruebas son efectuadas en las etapas iniciales del ciclo de vida antes de llegar a las pruebas de caja negra. Por otro lado, las pruebas de caja blanca son relativamente baratas porque se requiere menor comunicación y porque el análisis es más sencillo (la persona que detecta los defectos es regularmente la misma persona que hace las reparaciones. Además, se prueban menos objetos).

Niveles

La forma en que están organizadas las áreas clave dentro de un proceso de pruebas determina la “madurez” del proceso. Es obvio que no toda área clave tendrá la misma atención ni profundidad: cada proceso de pruebas tiene sus puntos fuertes y débiles. Con el fin de permitir una visión del status de cada área clave, el modelo proporciona **niveles** (de A a B a C). Como promedio, hay tres niveles por cada área.

Cada nivel superior (donde C es mayor que B, y B es mayor que A) es mejor que su nivel previo en términos de tiempo (más rápido), dinero (menor costo), y/o calidad (mejor). Al usar niveles podemos evaluar sin ambigüedades la situación prevaleciente del proceso de pruebas. También incrementa la posibilidad de recomendar mejoras graduales.

Cada nivel conlleva el cumplimiento de ciertos requisitos para el área clave. Los requisitos (= puntos de verificación o checkpoints) de un cierto nivel también incluyen los requisitos de niveles previos: un proceso de pruebas de nivel B satisface los requisitos de ambos niveles A y B. Si un proceso de pruebas no satisface los requisitos para el nivel A, se considera que se encuentra en el nivel más bajo y, por tanto, en un nivel indefinido para dicha área en particular.

A continuación se proporciona una descripción de los diferentes niveles de las áreas clave:

Niveles / Área Clave	A	B	C	D
Estrategia de pruebas	Estrategia de pruebas para una sola prueba	Estrategia de pruebas combinada para pruebas de caja negra	Estrategia de pruebas combinada para pruebas de caja negra y pruebas de caja blanca o evaluación	Estrategia de pruebas combinada para todas las pruebas y actividades de evaluación
Modelo del ciclo de vida	Planeamiento, diseño, ejecución	Planeamiento, preparación, diseño, ejecución, terminación		
Momento de involucrarse	Al terminar la especificación	Al inicio de la especificación	Al iniciar la definición de requerimientos	Al inicio del proyecto
Estimación y planeamiento	Estimación y planeamiento elemental	Estimación y planeamiento fundada estadísticamente		
Técnicas de diseño de pruebas	Técnicas informales	Técnicas formales		
Técnicas de pruebas estáticas	Pruebas en base a entradas	Listas de verificación (checklists)		
Métricas	Estadísticas del proyecto (producto)	Estadísticas del proyecto (proceso)	Estadísticas del sistema	Estadísticas de la organización
Herramientas de prueba	Herramientas de planeamiento y control	Herramientas de ejecución y análisis, y de pruebas	Automatización integral de pruebas	
Entorno de pruebas	Entorno administrado y controlado	Pruebas en el entorno más conveniente	Montaje de entorno a solicitud	
Entorno de oficina	Entorno de oficina adecuado y a tiempo			
Compromiso y motivación	Asignación de presupuesto y tiempo	Las pruebas integradas en la organización del proyecto	Ingeniería de pruebas	
Funciones de pruebas y capacitación	Gerente de pruebas y probadores	Soporte (metodológico, técnico, funcional), control	Aseguramiento de la calidad interna	
Alcance de la metodología	Específico al proyecto	Para toda la organización, genérica	Organización, optimización (Investigación y desarrollo)	
Comunicación	Comunicación interna	Comunicación del proyecto (defectos, cambios, control)	Comunicación en la organización	
Informes	Defectos	Progreso (status de pruebas y productos), actividades (costos + tiempo, metas), defectos con prioridades	Riesgo y consejo, incluyendo estadísticas	SPI consejo
Manejo de defectos	Administración interna de defectos	Administración extendida de defectos, informes flexibles	Administración de defectos del proyecto	
Administración de los elementos de prueba (testware)	Administración y control interno de los conceptos de pruebas	Administración y control externo de la base y de los objetos de pruebas	Testware reutilizable	Capacidad de rastreo: desde los requerimientos hasta los casos de prueba
Administración del proceso de pruebas	Planear, hacer	Planear, hacer, revisar, reaccionar	Comprobar, reaccionar en la organización	
Revisión estructurada	Técnicas de revisión estructurada	Estrategia de revisión estructurada		
Pruebas de caja blanca	Ciclo de vida: planear, diseñar, ejecutar	Técnicas de diseño de caja blanca	Estrategia de pruebas de caja blanca	

Puntos de verificación (checkpoints)

Con el fin de determinar los niveles, el modelo TPI se basa en un instrumento de medición objetiva. Los requisitos para cada nivel están definidos en forma de puntos de verificación (checkpoints): son preguntas que necesitan ser respondidas afirmativamente para poder calificar a dicho nivel. A partir de las listas de verificación se puede evaluar un proceso de pruebas y se puede determinar para cada área clave el nivel alcanzado. A medida que se considera mejorada cada área clave, los puntos de verificación son acumulables: para poder clasificar para el nivel B, el proceso de pruebas necesita responder afirmativamente tanto a los puntos de verificación del nivel B como del nivel A.

Matriz de madurez de pruebas

Después de determinar los niveles para cada área clave, se debe dirigir la atención hacia cuáles son los pasos de mejora que hay que realizar. Esto se debe a que no todas las áreas clave y niveles tienen la misma importancia. Por ejemplo, una buena estrategia de pruebas (nivel A del área clave "estrategia de pruebas") es más importante que una descripción de la metodología de pruebas utilizada (nivel A del área clave "alcance de la metodología"). Además de estas prioridades, existe una dependencia entre los niveles de diferentes áreas clave. Antes de poder recibir estadísticas de los defectos encontrados (nivel A del área clave "métricas"), el proceso de pruebas tiene que calificar para el nivel B del área clave "manejo de defectos". También hay dependencias entre muchos niveles y áreas clave.

Por lo tanto, todos los niveles y áreas clave están interrelacionados en una **matriz de madurez de pruebas**. Se ha concebido como una buena manera de expresar las prioridades internas y dependencias entre niveles y áreas clave. El eje vertical de la matriz indica áreas clave, el eje horizontal de la matriz muestra escalas de madurez. En la matriz, cada nivel está relacionado con cierta escala de madurez de pruebas. Resultando así 13 escalas de madurez de pruebas. Las celdas abiertas entre diferentes niveles no tienen significado por sí mismas, pero indican que el lograr una mayor madurez para un área clave está relacionado con la madurez de otras áreas clave. No existe graduación entre niveles: mientras que un proceso de pruebas no esté clasificado enteramente como nivel B, permanecerá en el nivel A.

Escala / Área Clave	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Estrategia de pruebas		A					B				C		D	
Modelo del ciclo de vida		A			B									
Momento de involucrarse			A				B				C		D	
Estimación y planeamiento				A							B			
Técnicas de diseño de pruebas		A		B										
Técnicas de pruebas estáticas					A		B							
Métricas						A			B			C		D
Herramientas de prueba					A			B			C			
Entorno de pruebas				A				B						C
Entorno de oficina				A										
Compromiso y motivación		A				B						C		
Funciones de pruebas y capacitación				A			B			C				
Alcance de la metodología					A						B			C

Escala / Área Clave	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Comunicación			A		B							C		
Informes		A			B		C					D		
Manejo de defectos		A				B		C						
Administración de los elementos de prueba (testware)			A			B				C				D
Administración del proceso de pruebas		A		B								C		
Revisión estructurada							A			B				
Pruebas de caja blanca					A		B		C					

El propósito principal de la matriz es mostrar los puntos fuertes y débiles del actual proceso de pruebas y ofrecer una ayuda a la hora de determinar la prioridad de las acciones de mejora. Una matriz con datos reales ofrece a todos los participantes una visión clara de la situación actual del proceso de pruebas. Además, la matriz ayuda a definir y seleccionar propuestas de mejora. La matriz se maneja de izquierda a derecha, de tal manera que las áreas claves poco maduras sean mejoradas en primer lugar. Como consecuencia de la dependencia entre niveles y áreas clave, la práctica nos ha demostrado que los "campeones aislados" (áreas clave con alta escala de madurez, rodeadas de áreas clave con escalas de madurez media o baja) no son una buena inversión. Por ejemplo, ¿de qué sirve tener una gestión de los defectos muy avanzada, si no se usa para realizar análisis e informes? Siempre que no se aparten del modelo, se permiten desviaciones, pero deberán existir razones de peso que lo justiquen.

En el ejemplo siguiente, el proceso de pruebas no clasifica para el nivel mas bajo del área clave "estrategia de pruebas" (nivel < A), la organización está trabajando conforme a un modelo de ciclo de vida (nivel A) y el personal que realiza las pruebas queda involucrado hasta el momento en que concluyan las especificaciones (nivel A).

Escala / Área Clave	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Estrategia de pruebas		A					B				C		D	
Modelo del ciclo de vida		A			B									
Momento de involucrarse			A				B				C		D	
etc.														

Partiendo en este caso de la matriz, se pueden discutir algunas mejoras. En este ejemplo, se elige una combinación de "estrategia de pruebas" para pruebas de caja negra (=> nivel B) y de un "modelo de ciclo de vida" (=> nivel B). En este momento, la involucración temprana no se considera relevante. La situación deseada está representada en la matriz siguiente:

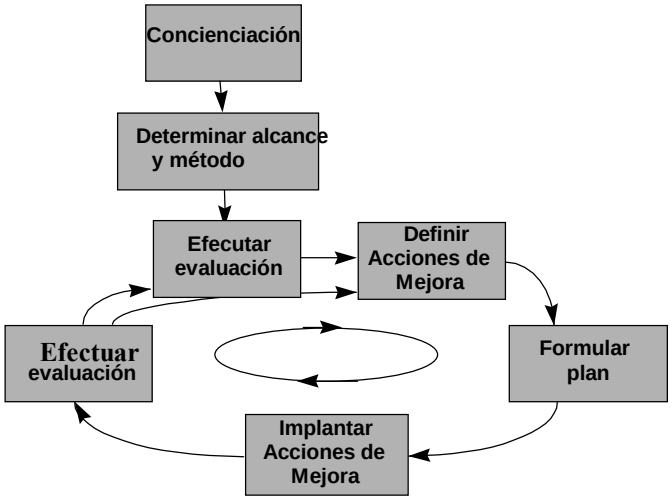
Escala / Área Clave	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Estrategia de pruebas		A					B				C		D	
Modelo del ciclo de vida		A			B									
Momento de involucrarse			A				B				C		D	
etc.														

Sugerencias de mejora

Las acciones de mejora pueden definirse en función de los niveles superiores que se deseen alcanzar. Para alcanzar un nivel superior, los puntos de verificación proporcionan gran ayuda. Además de éstos, el modelo tiene otros medios de soporte para la mejora del proceso de pruebas: Las "sugerencias de mejora", que son diferentes tipos de ideas o consejos que ayudan a alcanzar un cierto nivel de madurez de pruebas. A diferencia de los puntos de verificación, no es obligatorio usar las sugerencias de mejora. Cada nivel está provisto de varias sugerencias de mejora.

APLICACIÓN DEL MODELO TPI

El proceso de mejora de pruebas es similar a cualquier otro proceso de mejora. La figura siguiente muestra las diferentes actividades de un proceso de mejora. Estas actividades se van a comentar a continuación, poniendo especial atención a aquellas áreas en las que se puede ser utilizar el modelo TPI.



Concienciación

La primera actividad de un proceso de mejora de pruebas es tomar conciencia de la necesidad de mejorar el proceso. Hablando genéricamente, la razón para mejorar el proceso de pruebas es que existen una serie de problemas referentes a las pruebas. Hay que resolver dichos s problemas y una mejora del proceso de pruebas se ve como la solución. Esta concienciación también implica que las partes acuerdan mutuamente las líneas generales y se comprometen al proceso de cambio. El compromiso no sólo debe lograrse al inicio del proceso de cambio, sino que debe permanecer a lo largo de todo el proyecto. Esto requiere de un esfuerzo continuo.

Determinar el alcance y método

Determinamos cuáles son las metas y el alcance de la mejora: ¿Deberíamos realizar las pruebas con mayor rapidez, a menor costo o mejor?, ¿Cuáles son los procesos de prueba susceptibles de mejorar?, ¿De cuánto tiempo disponemos para hacer la mejora y cuánto esfuerzo estamos dispuestos a dedicarle?

Efectuar la evaluación

En esta actividad se obtiene una evaluación de la situación actual. El uso del modelo TPI es una parte importante de la evaluación porque ofrece un marco de referencia que enumera los puntos fuertes y débiles del proceso de pruebas. Basado en entrevistas y documentación, los niveles por área clave del modelo TPI son examinados al utilizar los puntos de verificación, y se determina cuáles son los puntos que se han alcanzado, y cuáles no se han alcanzado, o solo se han alcanzado parcialmente. La matriz de madurez de pruebas se utiliza aquí para ofrecer una visión completa de la situación en que se encuentra el proceso de pruebas. Aquí se mostrarán los puntos fuertes y débiles del proceso de pruebas en forma de niveles asignados a las áreas clave, así como su posición relativa en la matriz.

Definir acciones de mejora

Las acciones de mejora se definen a partir de los objetivos de mejora establecidos, así como de los resultados de la evaluación. Estas acciones se determinan de tal manera que sea posible ir mejorando paso a paso.

El modelo TPI ayuda a establecer estas acciones de mejora. Los niveles de las áreas clave y la matriz de madurez de pruebas ofrecen algunas posibilidades para definir pasos de mejora de modo gradual. Dependiendo de los objetivos, el alcance, el tiempo disponible y los resultados de la evaluación, se puede decidir iniciar mejoras para una o más áreas clave. Para cada área clave seleccionada se puede decidir ir al siguiente nivel o, en casos especiales, incluso a un nivel superior. Adicionalmente, el modelo TPI ofrece un gran número de sugerencias de mejora que ayudan a alcanzar niveles superiores.

Formular el plan

Se diseña un plan detallado para implantar (una parte de) las acciones de mejora a corto plazo. En este plan se determinan los objetivos y se indican cuáles son las mejoras que deben implantarse y en qué momento debe hacerse con el fin de lograr los objetivos. El plan se dirige tanto a las actividades relacionadas con el contenido de las mejoras del proceso de pruebas como a las actividades generales necesarias para realizar el proceso de cambio en la dirección adecuada.

Implantar acciones de mejora

Se realiza el plan. Debido a que durante esta actividad, las consecuencias del proceso de cambio tienen un mayor impacto, hay que prestar mucha atención a la comunicación. Sin duda, puede haber cierta resistencia al cambio, por lo que deberá afrontarse y discutirse abiertamente.

Se debe medir hasta qué punto se han ejecutado las acciones y si han resultado un éxito. Una forma de hacerlo es mediante la llamada “auto-evaluación”, en la que se aplica el modelo TPI con el fin de determinar rápidamente el progreso obtenido. Aquí, las personas involucradas revisan su propio proceso de pruebas usando el modelo TPI.

Otra parte vital de esta fase es la consolidación. Se debe evitar que las acciones de mejora implantadas sean experimentadas únicamente una vez.

Efectuar evaluación

¿Hasta qué punto las acciones implantadas han logrado los resultados esperados? En esta fase el objetivo es ver en qué medida las acciones fueron implantadas con éxito, así como evaluar en qué medida se alcanzaron los objetivos iniciales. A partir de estas observaciones se adoptará la decisión sobre la continuación del proceso de cambio.

CAPÍTULO 2. SITUACIÓN ACTUAL

2.1. INTRODUCCIÓN

En este capítulo se describen algunas de las metodologías de administración de proyectos y desarrollo de software existentes en el mercado, describiendo de forma muy global cada una de ellas.

2.2. EL VALOR DE LA ADMINISTRACIÓN DE PROYECTOS

Existen algunas compañías que han construido su reputación alrededor de su habilidad para administrar proyectos de manera efectiva. Sin embargo, para la gran mayoría de las organizaciones esa reputación es más bien marginal.

Para poder llegar a establecer el valor dentro de la organización, se deben evaluar si se posee alguna de las siguientes características:

- Los proyectos terminan después de la fecha prometida, o cuestan más de lo originalmente presupuestado o bien no cumplen con los requerimientos de funcionalidad señalados por el cliente.
- Existen procesos y técnicas pobremente estandarizados y son usados de manera inconsistente por los administradores de proyecto.
- La administración de proyectos es reactiva y no se ve como generadora de valor.
- El tiempo requerido para la gestión del proyecto no se incluye en el plan de trabajo, dado que es considerado como un gasto general.
- Los proyectos no son exitosos debido a la falta de planificación y gestión, por lo que existe gran estrés y es necesario trabajar tiempo extra a lo largo del ciclo de vida de todo el proyecto.

Una buena disciplina de administración de proyectos es la forma en que una organización se puede sobreponer a estos problemas. Tener habilidades en gestión de proyectos, no quiere decir que no se tendrán problemas. No significa que los riesgos simplemente desaparezcan, o que no haya sorpresas. El valor de una buena práctica de administración de proyectos es que la organización contará con un proceso estándar para lidiar con todo este tipo de contingencias.

2.2.1. Administración de proyectos: ciencia o arte

Los procesos y técnicas de administración de proyectos son usados para coordinar recursos con el fin de alcanzar resultados predecibles. Sin embargo, se debe entender de antemano que la administración de proyectos no es enteramente una ciencia, por lo que nunca existe una garantía de que haya resultados exitosos. Dado que la ejecución de proyectos involucra gente, siempre existirá un factor de complejidad e incertidumbre que no podrá controlarse totalmente.

La administración de proyectos es una ciencia en lo que se refiere al uso de procesos probados y repetibles y a las técnicas que permiten alcanzar resultados exitosos. Pero es un arte debido a que tiene mucho que ver con relacionarse y manejar gente y esto requiere de habilidades intuitivas que han de aplicarse a situaciones que varían y a menudo son

totalmente únicas de proyecto a proyecto. Una buena metodología de gestión de proyectos proporciona el esquema de trabajo, los procesos, normas y técnicas para manejar a la gente y la cantidad de trabajo asociado; por lo que ésta incrementa las probabilidades de tener éxito, y en consecuencia proporciona valor a la organización, al proyecto y al Gerente del Proyecto.

2.2.2. Propuesta de valor de la dirección de proyectos

La propuesta de valor de la dirección de proyectos plantea algo como siguiente: toma tiempo y esfuerzo administrar proyectos de manera proactiva. Este costo es más que recuperado a lo largo del ciclo de vida del proyecto debido a que:

- **Se finalizan proyectos más rápido y a un menor costo:** Uno de los mayores beneficios de usar una metodología común es el valor de la reutilización. Una vez que los procesos, procedimientos y plantillas son creados, éstos pueden ser usados (quizás con pequeñas modificaciones) en todos los proyectos futuros. El resultado de esto es un menor tiempo para iniciar proyectos, una menor curva de aprendizaje para los miembros del equipo de trabajo así como ahorros de tiempo al no tener que reinventar procesos y plantillas desde cero en cada proyecto.

- **Ahorros de tiempo y costo con una administración proactiva del alcance:** Gran cantidad de proyectos tienen dificultades en la gestión del alcance, lo que resulta en esfuerzos y costos adicionales para el proyecto. Al tener mejores procesos de administración de proyectos se podrá gestionar el alcance de manera más efectiva.

- **Mejores soluciones que se "ajustan" a la primera a través de una mejor planeación:** Gran cantidad de proyectos experimentan problemas debido a que hay brechas entre las expectativas del cliente y lo que el proyecto entrega. El uso de una metodología resulta en una mejor planeación del proyecto, la cual le da al equipo y al patrocinador del proyecto la oportunidad de asegurar que existe acuerdo en cuanto a los entregables mayores que serán producidos por el proyecto.

- **Resolución de problemas más rápido:** Algunos equipos dedican mucho tiempo y energía enfrentando problemas debido a que no saben como deben iniciar la solución de éstos. El tener un proceso de gestión proactiva de incidentes ayuda a asegurar que los problemas son resueltos tan rápido como es posible.

- **Resolver riesgos futuros antes de que se materialicen en problemas:** El uso de procesos adecuados de gestión de riesgos va a resultar en una identificación oportuna de problemas potenciales y en la definición de actividades contingentes antes de que éstos ocurran.

- **Comunicación y manejo de expectativas con clientes, miembros del equipo y grupos de interés más efectiva:** Muchos de los problemas que se presentan en un proyecto pueden ser evitados a través de una comunicación proactiva y multifacética. Además, muchos de los conflictos que surgen en el proyecto no son resultado de un problema específico sino de sorpresas.

- **Creación de productos y servicios de mayor calidad la primera vez:** Los procesos de gestión de calidad ayudan al equipo a entender las necesidades del cliente en términos de calidad. Una vez que esas necesidades son definidas, el equipo puede implementar controles de calidad y técnicas que garanticen la calidad que permita alcanzar las expectativas del cliente.

- **Gestión financiera mejorada:** Este es el resultado de una mejor definición del proyecto, mejores estimaciones, un proceso de presupuestación más formal y un mejor seguimiento de los costos reales del proyecto contra el presupuesto. Todo este rigor resulta en una mejor predictibilidad financiera y control.

- **Poner un alto a los "malos" proyectos con mayor prontitud.** Los "malos" proyectos son aquellos en donde la justificación costo-beneficio no tiene ya sentido. Un proyecto puede haber iniciado con una clara justificación costo-beneficio. Sin embargo, si el proyecto toma demasiado tiempo y ha excedido su presupuesto, puede alcanzar el umbral en que el negocio ya no es válido. La gestión efectiva de proyectos permite identificar este tipo de situación oportunamente de manera que se puedan tomar mejores decisiones para replantear el alcance o para cancelar el proyecto.

- **Mayor enfoque en las métricas y un proceso de toma de decisiones basado en hechos:** Uno de los aspectos más sofisticados de las metodologías es que éstas proporcionan una guía para facilitar la recolección de métricas (medidas). Las métricas proporcionan información que ayuda a determinar que tan efectivo y eficiente está siendo el desempeño del equipo y el nivel de calidad de los entregables. Las métricas también proporcionan la información necesaria para validar si se fue o no exitoso.

- **Ambiente laboral mejorado:** Si los proyectos son más exitosos, se encontrarán beneficios adicionales asociados al equipo de proyecto. Los clientes tendrán mayor involucración, el equipo tendrá más sentido de propiedad sobre el proyecto, la moral será mejor y los miembros del equipo se comportarán con mayor profesionalismo y auto-confianza. Esto debe tener sentido. La gente que trabaja en proyectos con problemas tiende a ser infeliz. Por otra parte, la gente que trabaja en proyectos exitosos, tiende a sentirse más satisfecha con su trabajo y con ellos mismos.

La gente que se queja de que la gestión de proyectos representa mucho costo administrativo, olvida el hecho de que el proyecto enfrentará incidentes. ¿Se quiere resolver proactivamente estos o pensar en ellos conforme se vaya avanzando en el proyecto? El proyecto enfrentará riesgos potenciales ¿Se quiere intentar resolverlos antes de que se presenten o hasta que éstos sean problemas que amenazan al proyecto? ¿Se buscará establecer proactivamente canales de comunicación o se quiere lidiar con los conflictos y la incertidumbre causados por la falta de información del proyecto? ¿Se manejará realizará una gestión adecuada del alcance o se prefiere enfrentar problemas de retrasos y sobrecostos ocasionados por hacer mas trabajo del que se había presupuestado? ¿Se integrará la calidad dentro del proceso o se prefiere corregir los problemas de una mala calidad después, cuando resolverlos es más caro y complicado?

Las características del proyecto no van a cambiar si se usa un proceso formal de administración de proyectos o no. Lo que cambia es la forma en que los eventos son manejados cuando el proyecto está en curso.

2.2.3. ¿Por que usar una administración formal de proyectos?

Generalmente, se cree que las organizaciones que siguen buenos procesos, son más exitosas que aquellas que no lo hacen. Las organizaciones que tienen buenos procesos y los siguen, a menudo son conocidas como “organizaciones orientadas a procesos”. Estas organizaciones procesan más trabajo y el trabajo que conducen tiende a ser más valioso. También tienen sistemas organizacionales para ayudar a todos sus integrantes a ser más exitosos, incluyendo a los gerentes de proyecto. Hay diversas formas de evaluar que también la organización se apega a procesos estándares.

Después de leer esta sección, puede surgir la pregunta de por qué no todo mundo usa una buena práctica de gestión de proyectos, sus técnicas y sus procesos, o podemos preguntarnos ¿por qué no lo las estoy usando? Habitualmente, esto se debe principalmente a varias razones:

- **Una buena administración de proyectos requiere inversión adelantada de tiempo y esfuerzo:** Mucha gente se considera a sí misma como “hacedores”. Pueden no estar tan satisfechos con sus habilidades de planificación. Muchas veces existe una tendencia a discutir un problema y correr a solucionarlo. Esto funciona bien cuando se tiene un requerimiento de cambio de cinco horas pero no es lo mismo cuando esta solicitud implica cinco mil horas de esfuerzo al proyecto. Se debe resistir la urgencia de poner manos a la obra. El proyecto terminará antes si se planifica apropiadamente por adelantado, y después se debe ser disciplinado para gestionarlos efectivamente.

- **La organización no está comprometida:** Es difícil ser un buen Gerente del Proyecto en una organización que no valora las habilidades que implica la administración de proyectos. Por ejemplo, si alguien se toma el tiempo para generar el documento de definición de proyecto y el cliente pregunta por que se perdió el tiempo haciéndolo, quizás el proceso de planeación del próximo proyecto no vaya a ser muy satisfactorio para esa persona. Para que la organización entera sea más efectiva debe apoyar el uso de un proceso común de gestión de proyectos.

- **No se tienen las habilidades adecuadas:** Se puede notar que la falta de procesos de administración de proyectos no es un asunto de voluntad, sino de habilidades. Algunas veces se le pide a la gente que conduzca un proyecto sin la capacitación o la experiencia necesaria. En esos casos, luchan sin las herramientas adecuadas o el entrenamiento apropiado para gestionar el proyecto adecuadamente. Su organización puede que tampoco tenga una Oficina de Administración de Proyectos (PMO) u otra organización responsable de desplegar estas habilidades de gestión.

- **La alta dirección piensa en la Administración de Proyectos como una herramienta:** Al hablar de la administración de proyectos con algunos gerentes, inicialmente tienden a pensar que se trata de implementar una herramienta que permita que la organización cuente con mejores administradores de proyecto. De hecho, sí se tratara de una herramienta, se tendría mas suerte al tratar de convencerlos de su valor. Aun cuando algunos aspectos de la administración de proyectos, como la creación y administración del plan de trabajo, pueden usar una herramienta, no es ahí donde está el valor de administrar proyectos. El valor está en la utilización disciplinada de procesos sólidos y consistentes.

- **El concepto puede haberse “quemado” (o enterrado) en el pasado:** Cuando se empieza a hablar de procesos, mejores prácticas y plantillas, algunos gerentes inmediatamente empiezan a pensar en gastos, retrasos y papeleo. No logran conectarse inmediatamente con el valor que una metodología conlleva. Una crítica común de la metodología es que es engorrosa, involucra un uso de papel excesivo y

quita concentración del trabajo encomendado. Algunas veces estas críticas se refieren a problemáticas legítimas provocadas por no adecuar el uso de la metodología a las características del proyecto. Por ejemplo, si se pide al líder de proyecto que produzca un documento de definición del proyecto de 15 hojas, aún cuando el proyecto requiere 250 horas de esfuerzo, Sin embargo esto no es un problema de la metodología sino de una mala aplicación de la misma.

- **Existe aversión al control en el equipo de trabajo:** A una gran cantidad de gente le gusta hacer su trabajo de manera creativa y con un mínimo de supervisión. Este tipo de gente teme que el uso de técnicas formales de administración de proyectos resulte en un riguroso control que eliminará la creatividad y diversión del trabajo. Hasta cierto punto tienen razón. Sin embargo, el uso de procesos y procedimientos comunes eliminaran algo de la creatividad en áreas donde probablemente la organización no desee ser creativa. No se necesita mucha creatividad al enfrentar cambios de alcance, por ejemplo. Solo es necesario apegarse a los procesos estándares que ya existen en la organización.

- **Existe el temor por parte de la gerencia media o alta dirección a perder el control:** Si realmente se desea implementar efectivamente una disciplina de administración de proyectos en la organización, es necesario otorgar un cierto nivel de control y autoridad al Gerente de Proyecto. En algunas organizaciones, la gerencia media, no desea perder ese control. Quizás quieren que los administradores de proyecto coordinen los proyectos, pero la gerencia media quiere tomar todas las decisiones y ejercer todo el control. La administración de proyectos formal no será posible en organizaciones en donde este temor prevalezca.

Algunos de estos temores son naturales y lógicos, mientras que otros son emocionales e irracionales. Aunque estas pueden ser razones para estar indeciso acerca de usar procesos formales de administración de proyectos, éstas deben ser superadas. El tema central en administración de proyectos es –Si el resultado de la administración de proyectos fuera que éstos se desarrollaran más despacio, costaran más y tuvieran mala calidad, no tendría sentido usarla. De hecho, todo lo contrario es cierto. El usar técnicas y procesos sanos de administración de proyectos dará una mayor probabilidad de que el proyecto finalice a tiempo, dentro de presupuesto y con aceptable nivel de calidad.

Dicho lo anterior, cuando se use un proceso de administración de proyectos, se debe ser inteligente. No construya procesos para proyectos de 10 millones de dólares si el proyecto solo cuesta 5 mil dólares. Además se deben considerar todos los aspectos respecto a como manejar un proyecto, las características particulares del proyecto y en función de ello, construir los procesos específicos que se ajusten a éste lo mejor posible.

2.2.4. Opciones para obtener una metodología

Para implementar satisfactoriamente una metodología de administración de proyectos, primero se debe convencer uno mismo de que existe valor si los procesos son aplicados y utilizados correctamente. De hecho, todos los proyectos usan una mezcla de metodología de procesos, procedimientos y plantillas. Si se cree que no se tiene ninguna, esto significa en realidad que se tiene una metodología pobre e informal.

Si se necesita una buena metodología de administración de proyectos, existen dos fuentes principales en donde obtenerla:

1. **Construir una:** Se puede construir una metodología adaptada a la propia organización que refleje perfectamente la filosofía y mejores prácticas de la compañía. Una gran cantidad de compañías continúan haciendo esto hoy en día.

2. **Comprar una:** Si se construye una metodología, es probable que exista algún grado de sorpresa al descubrir que, a final de cuentas, ésta se ve muy similar a otras metodologías de gestión de proyectos que la gente usa. No importa como se estructure, siempre será necesario planificar, construir el plan de trabajo, gestionar alcance, riesgos e incidentes, comunicar, etc. En consecuencia, una gran cantidad de empresas eligen la opción de comprar una licencia o una metodología preexistente. En cualquier caso éstas habitualmente contienen todo lo que una organización necesita para una gestión de proyectos exitosa.

También existe la opción híbrida de comprar una metodología y adecuarla a las necesidades específicas de la organización. Esto de alguna forma, proporcional los beneficios de la opción 1, a la vez que toma menos tiempo, lo que representa la mayor ventaja de la opción 2.

2.3. METODOLOGÍAS EXISTENTES

Cada metodología tiene su propia manera de asentar los procesos, procedimientos, prácticas de trabajo y plantillas requeridas para manejar exitosamente los proyectos. Si se observa detalladamente, se descubren gran cantidad de similitudes. Asimismo están presentes las diferencias, no tanto por desacuerdos importantes, sino por el énfasis que se da en una u otra. Por ejemplo, hay algunas metodologías que incluyen la recopilación de requerimientos de negocio como parte del proceso de administración de proyectos. Si un gran componente de trabajo es dividido en múltiples fases, un proyecto completo podría ser la fase de análisis, y otro proyecto sería la fase de diseño y construcción. En el segundo proyecto, no habría ninguna necesidad de recopilar requerimientos de negocio para nada.

A continuación se describirán las metodologías más comunes en el mercado.

2.3.1. Project Management Body of Knowledge (PMBOK)

Una de descripción estándar de procesos para administración de proyectos, es “el cuerpo de conocimiento de la administración de proyectos” (PMBOK, por sus siglas en inglés), que es el estándar publicado por el Instituto de Administración de Proyectos (PMI, por sus siglas en inglés). El PMBOK, contiene gran cantidad de información valiosa, proporcionando un fundamento básico para entender como funciona el trabajo como proyecto, pero no necesariamente es una metodología que puede tomarse y ser implantada para administrar un proyecto directamente. Por ejemplo, hay información, pero no procedimientos. Hay definiciones, pero no necesariamente prácticas de trabajo o técnicas. Igualmente se señalan las salidas o entregables de cada proceso, pero no proporciona formatos y plantillas de éstos.

El PMBOK es una colección de procesos y áreas de conocimiento generalmente aceptadas como las mejores prácticas dentro de la administración de proyectos. El PMBOK en un estándar reconocido internacionalmente (IEEE Standard 1490-2003) que provee los fundamentos de la gestión de proyectos que son aplicables a un amplio rango de proyectos, incluyendo construcción, software, ingeniería, etc.

El PMBOK reconoce 5 procesos básicos y 9 áreas de conocimiento comunes a casi todos los proyectos. Los conceptos básicos son aplicables a proyectos, programas y operaciones. Los cinco grupos de procesos básicos son:

1. Inicio
2. Planificación
3. Ejecución
4. Control y monitoreo
5. Cierre

Los procesos se traslapan e interactúan a través de un proyecto o fase. Los procesos son descritos en términos de: entradas (documentos, planes, diseños, etc.), herramientas y técnicas (mecanismos aplicados a las entradas) y salidas (documentos, productos, etc.). Las nueve áreas del conocimiento mencionadas en el PMBOK son:

1. Gestión de la integración del proyecto
   - 1.1 Desarrollar el acta de constitución del proyecto
   - 1.2 Desarrollar el enunciado del alcance del proyecto (preliminar)
   - 1.3 Desarrollar el plan de administración del proyecto
   - 1.4 Dirigir y gestionar la ejecución del proyecto
   - 1.5 Supervisar y controlar el trabajo del proyecto
   - 1.6 Control integrado de cambios
   - 1.7 Cerrar proyecto
2. Gestión del alcance del proyecto
   - 2.1. Planificación del alcance
   - 2.2. Definición del alcance
   - 2.3. Crear la estructura de desglose del trabajo
   - 2.4. Verificación del alcance
   - 2.5. Control del alcance
3. Gestión del tiempo del proyecto
   - 3.1. Definición de las actividades
   - 3.2. Establecimiento de la secuencia de actividades
   - 3.3. Estimación de recursos de las actividades
   - 3.4. Estimación de la duración de las actividades
   - 3.5. Desarrollo del cronograma
   - 3.6. Control del cronograma
4. Gestión de los costos del proyecto
   - 4.1. Estimación de costos
   - 4.2. Preparación del presupuesto de costos
   - 4.3. Control de costos
5. Gestión de la calidad del proyecto
   - 5.1. Planificación de calidad
   - 5.2. Realizar aseguramiento de calidad
   - 5.3. Realizar control de calidad

6. Gestión de los recursos humanos del proyecto
   - 6.1. Planificación de los recursos humanos
   - 6.2. Adquirir el equipo del proyecto
   - 6.3. Desarrollar el equipo del proyecto
   - 6.4. Gestionar el equipo del proyecto

7. Gestión de las comunicaciones del proyecto
   - 7.1. Planificación de las comunicaciones
   - 7.2. Distribución de la información
   - 7.3. Informar el rendimiento
   - 7.4. Gestionar a los interesados

8. Gestión de los riesgos del proyecto
   - 7.1. Planificación de la gestión de riesgos
   - 7.2. Identificación de riesgos
   - 7.3. Análisis cualitativo de riesgos
   - 7.4. Análisis cuantitativo de riesgos
   - 7.5. Planificación de la respuesta a los riesgos
   - 7.6. Seguimiento y control de riesgos

9. Gestión de las adquisiciones del proyecto
   - 9.1. Planificar las compras y adquisiciones
   - 9.2. Planificar la contratación
   - 9.3. Solicitar respuestas de vendedores
   - 9.4. Selección de vendedores
   - 9.5. Administración del contrato
   - 9.6. Cierre del contrato

2.3.2. Projects IN Controlled Environments (PRINCE2)

PRINCE2 es una metodología de administración de proyectos que fue creada para uso del gobierno del Reino Unido. También es usada en el sector privado y se ha venido haciendo popular en muchas naciones europeas. El proceso de PRINCE2 está dividido en 8 procesos. Tres de ellos (ideando el proyecto, iniciando el proyecto y planeación) están relacionados con la planeación del proyecto. Tanto como ideando el proyecto como iniciando el proyecto ocurren al principio de la metodología PRINCE2, pero la planeación es un proceso continuo que ocurre durante la vida del proyecto.

PRINCE2 dedica tres de los procesos restantes a la ejecución del proyecto. La dirección del proyecto en PRINCE2 es desarrollada por un Comité de Dirección del Proyecto y no se involucra con las actividades diarias del Gerente de Proyecto.

Los 8 grupos de procesos son:

1. Ideando un proyecto (SU: Starting up a project)
   - SU1: Nombrando un Comité de Dirección del proyecto y al Gerente de Proyecto
   - SU2: Diseño del equipo de proyecto
   - SU3: Designar el equipo de trabajo del proyecto

SU4: Preparación del informe del proyecto
SU5: Definición del enfoque del proyecto
SU6: Planeación de la fase de inicio

2. Planificación (PL: Planning)

PL1: Diseñando un plan
PL2: Definiendo y analizando productos
PL3: Identificando acciones y dependencias
PL4: Estimación
PL5: Programando las actividades
PL6: Analizando los riesgos
PL7: Completando el plan

3. Iniciando el proyecto (IP: Initiating a project)

IP1: Planeación de la calidad
IP2: Planeación del proyecto
IP3: Refinación del plan de negocio y riesgos
IP4: Preparación de los controles del proyecto
IP5: Creación de los archivos del proyecto
IP6: Ensamble del proyecto documento de inicio (PID)

4. Dirigiendo el proyecto (DP: Directing a project)

DP1: Autorización del inicio
DP2: Autorización del proyecto
DP3: Autorización de una fase o el plan de excepción
DP4: Dando dirección ad-hoc al proyecto
DP5: Confirmación del cierre de proyecto

5. Controlando una fase (CS: Controlling a stage)

CS1: Autorización de un paquete de trabajo
CS2: Evaluación del progreso
CS3: Capturando los incidentes del proyecto
CS4: Examinando los incidentes del proyecto
CS5: Revisando el estatus de la fase
CS6: Reportando puntos críticos
CS7: Tomando acciones correctivas
CS8: Escalando los incidentes del proyecto
CS9: Recibiendo el paquete de trabajo concluido

6. Manejando la entrega del producto (MP: Managing product delivery)

MP1: Aceptando un paquete de trabajo
MP2: Ejecutando un paquete de trabajo
MP3: Entregando el paquete de trabajo

7. Manejando los límites de fase (SB: Managing stage boundaries)

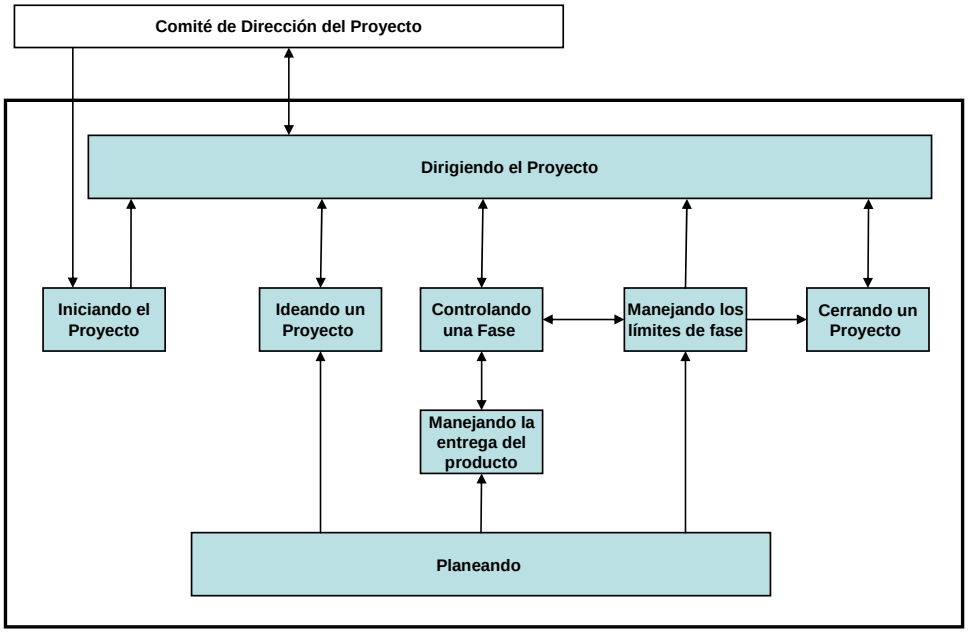
SB1: Planeando una fase
SB2: Actualizando el plan de proyecto

SB3: Actualizando el plan de negocio
SB4: Actualizando la bitácora de riesgos
SB5: Reportando el final de fase
SB6: Produciendo un plan de excepción

8. Cerrando el proyecto (CP: Closing a project)

CP1: Desmantelando un proyecto
CP2: Identificando acciones de seguimiento
CP3: Revisión de la evaluación del proyecto

A continuación se muestra gráficamente la estructura de la metodología PRINCE2:



2.3.3. Six Sigma

Six Sigma se refiere a una filosofía, meta o metodología usada para reducir el desperdicio y mejorar la calidad, los costos y el tiempo en que se desarrolla cualquier negocio. Sigma es una letra griega usada para indicar el monto de la variación o el nivel de defectos en un producto (un defecto es definido como cualquier cosa que provoca una insatisfacción del cliente o quejas acerca de las tolerancias aceptadas). Hoy en día, una organización típica debe estarse desempeñando en un nivel tres Sigma, lo que significa que experimentan un defecto por cada 16 oportunidades. Esto equivale aproximadamente a 67,000 defectos por millón de oportunidades. Una compañía con un desempeño mejor puede ubicarse en un nivel cuatro Sigma o un defecto por cada 160 oportunidades; lo cual no está mal pero continúan existiendo 6,000 defectos por millón. Un nivel de desempeño seis sigma equivale a 3.4 defectos por millón de oportunidades – no es perfecto, pero se acerca mucho.

En general Six Sigma es una filosofía que proporciona a las organizaciones una serie de procesos y herramientas estadísticas que conducen a un incremento en la rentabilidad y la calidad – ya sea que una compañía produzca bienes de consumo duradero o servicios. Six

Sigma es un proceso de largo plazo que busca la mejora continua. Estas mejoras no pueden alcanzarse por la reestructuración de la empresa o simplemente por gastar grandes cantidades de dinero. En lugar de ello, la calidad Six Sigma requiere perseverancia, enfoque y dedicación.

El nivel de calidad Six Sigma puede ser alcanzado a través de la combinación continua de proyectos sistemáticos y estructurados. Los proyectos son categorizados en:

- Proyectos de procesos transaccionales de negocio, que se extienden a lo largo de la organización.
- Proyectos tradicionales de mejora de calidad que resuelven problemas crónicos y que se extienden a diversos puntos dentro de una organización.

Una compañía, organización o proyecto que trata de alcanzar un nivel Six Sigma de calidad, debería de centrarse en el diseño de productos, servicios o procesos. Los principios Six Sigma pueden ser aplicados a muchas áreas, incluyendo producción, servicios administrativos o al cliente. Las técnicas Six Sigma ayudan a disminuir la variabilidad, la cual reduce el número de defectos y los costos de operación mientras que incrementa la capacidad efectiva. Las técnicas Six Sigma también pueden ser usadas para mejorar otros aspectos, como la lealtad de los clientes, que de paso ayuda a la razón de ser de la organización.

Dentro de los beneficios que se obtienen del Six Sigma están: mejoramiento de la rentabilidad y la productividad. Una diferencia importante con relación a otras metodología es la orientación al cliente.

El proceso Six Sigma se caracteriza por 5 etapas bien definidas:

- **Definir** el problema o el defecto
- **Medir** y recopilar datos
- **Analizar** datos
- **Mejorar**
- **Controlar**

Las metodologías son: DMAIC y DMADV

DMAIC: Definir, medir, analizar, mejorar y controlar
DMADV: Definir, medir, analizar, diseñar y verificar

D (DEFINIR)
En la fase de definición se identifican los posibles proyectos Six Sigma, que deben ser evaluados por la dirección para evitar la infrautilización de recursos. Una vez seleccionado el proyecto se prepara su misión y se selecciona el equipo más adecuado para el proyecto, asignándole la prioridad necesaria.

¿Qué procesos existen en su área? ¿De cuáles actividades (procesos) es usted el responsable? ¿Quién o quiénes son los dueños de estos procesos? ¿Qué personas interactúan en el proceso, directa e indirectamente? ¿Quiénes podrían ser parte de un equipo para cambiar el proceso? ¿Tiene actualmente información del proceso? ¿Qué tipo de información tiene? ¿Qué procesos tienen mayor prioridad de mejorarse? ¿Cómo lo definió o llegó a esa conclusión?

M (MEDIR)
La fase de medición consiste en la caracterización del proceso identificando los requisitos clave de los clientes, las características clave del producto (o variables del resultado) y los parámetros (variables de entrada) que afectan al funcionamiento del proceso y a las características o variables clave. A partir de esta caracterización se define el sistema de medida y se mide la capacidad del proceso.

¿Sabe quiénes son sus clientes? ¿Conoce las necesidades de sus clientes? ¿Sabe qué es critico para su cliente, derivado de su proceso? ¿Cómo se desarrolla el proceso? ¿Cuáles son los pasos? ¿Qué tipo de pasos compone el proceso? ¿Cuáles son los parámetros de medición del proceso y cómo se relacionan con las necesidades del cliente? ¿Por qué son esos los parámetros? ¿Cómo obtiene la información? ¿Qué tan exacto o preciso es su sistema de medición?

A (ANALIZAR)
En la fase, análisis, el equipo analiza los datos de resultados actuales e históricos. Se desarrollan y comprueban hipótesis sobre posibles relaciones causa-efecto utilizando las herramientas estadísticas pertinentes. De esta forma el equipo confirma los determinantes del proceso, es decir las variables clave de entrada o "pocos vitales" que afectan a las variables de respuesta del proceso.

¿Cuáles son las especificaciones del cliente para sus parámetros de medición? ¿Cómo se desempeña el proceso actual con respecto a esos parámetros? Muestre los datos. ¿Cuáles son los objetivos de mejora del proceso? ¿Cómo los definió? ¿Cuáles son las posibles fuentes de variación del proceso? Muestre cuáles y qué son. ¿Cuáles de esas fuentes de variación controla y cuáles no? De las fuentes de variación que controla ¿Cómo las controla y cuál es el método para documentarlas? ¿Monitorea las fuentes de variación que no controla?

I [IMPROVE] (MEJORAR)
En la fase de mejora el equipo trata de determinar la relación causa-efecto (relación matemática entre las variables de entrada y la variable de respuesta que interese) para predecir, mejorar y optimizar el funcionamiento del proceso. Por último se determina el rango operacional de los parámetros o variables de entrada del proceso.

¿Las fuentes de variación dependen de un proveedor? Si es así ¿Cuáles son?, ¿Quién es el proveedor? y ¿Qué está haciendo para monitorearlas y/o controlarlas? ¿Qué relación hay entre los parámetros de medición y las variables críticas? ¿Interactúan las variables críticas? ¿Cómo lo definió? Muestre los datos. ¿Qué ajustes a las variables son necesarios para optimizar el proceso? ¿Cómo los definió? Muestre los datos.

C (CONTROLAR)
Fase, control, consiste en diseñar y documentar los controles necesarios para asegurar que lo conseguido mediante el proyecto Six Sigma se mantenga una vez que se hayan implantado los cambios. Cuando se han logrado los objetivos y la misión se dé por finalizada, el equipo informa a la dirección y se disuelve.

Para las variables ajustadas ¿Qué tan exacto o preciso es su sistema de medición? ¿Cómo lo definió? Muestre los datos. ¿Qué tanto se ha mejorado el proceso después de los cambios? ¿Cómo lo define? Muestre los datos. ¿Cómo hace que los cambios se mantengan? ¿Cómo monitorea los procesos? ¿Cuánto tiempo o dinero ha ahorrado con los cambios? ¿Cómo lo está documentando? Muestre los datos.

2.3.4. Norma ISO 10006

La Organización Internacional de Estandarización (ISO, por sus siglas en inglés) es una organización especializada que promueve el desarrollo de estándares precisos para asegurar que productos, servicios y materiales a lo largo de las naciones afiliadas permanezcan consistentes. Esta estandarización ayuda a facilitar el intercambio internacional de bienes y servicios y a desarrollar la cooperación en actividades intelectuales, científicas, tecnológicas y económicas. Los resultados del trabajo técnico ISO son publicados como estándares internacionales.

La esperanza es que si una empresa usa procesos que resultan en el cumplimiento de una certificación ISO, los productos producidos por ese proceso cumplirán los estándares mínimos para todas las naciones en la materia específica.

Los estándares internacionales para la dirección de proyectos están reflejados en la norma ISO 10006, y son similares en naturaleza a lo que plantea el PMBOK del Instituto de Dirección de Proyectos (PMI por sus siglas en inglés), aunque la norma ISO es menos extensa. El documento ISO10006 en sí mismo es escrito a un nivel muy alto.

El siguiente cuadro describe la norma ISO a un nivel muy alto, la fuente es el documento de estándares ISO 10006.

Es importante señalar que la norma ISO 10006 tiene algunas sub-cláusulas y sub-sub-cláusulas adicionales que proporcionan mayor perspectiva y guía desde el punto de vista organizacional. Sin embargo, la tabla que se presenta más adelante representa los procesos ISO 10006 que se refieren específicamente a los procesos relacionados con los proyectos.

ISO 10006				
Cláusula	**Sub-cláusula**	**Sub-sub-cláusula**	**Proceso**	**Descripción del proceso**
5. Responsabilidad de la dirección	5.2 Proceso estratégico	5.2	Estratégico	Un proceso de establecimiento de directrices, que incluye la planeación e implementación del sistema de gestión de calidad con base en los principios de gestión de calidad.
6. Gestión de recursos	6.1 Procesos relacionados con la gestión de recursos	6.1.2	Planeación de recursos	Identificación, estimación, calendarización y asignación de todos los recursos relevantes.
		6.1.3	Control de recursos	Comparación del uso real de recursos versus las necesidades de recursos y tomar las acciones conducentes en caso de ser necesario.
	6.2 Procesos relacionados con el personal	6.2.2	Definición de la estructura de organización del proyecto	Definición de la estructura organizacional del proyecto ajustada para que cumpla con las necesidades del proyecto, incluyendo la identificación de roles y estableciendo la autoridad y responsabilidad correspondientes.
		6.2.3	Asignación del personal	Selección y asignación del personal suficiente con el conjunto de competencias necesarias para el proyecto.
		6.2.4	Desarrollo del equipo	Desarrollo de las habilidades y competencias individuales y de equipo para mejorar el desempeño del proyecto.

ISO 10006				
Cláusula	**Sub-cláusula**	**Sub-sub-cláusula**	**Proceso**	**Descripción del proceso**
7. Creación del producto	7.2 Procesos relacionados con interdependencias	7.2.2	Inicio del proyecto y desarrollo del plan de trabajo	Evaluación de los requerimientos del cliente y otros grupos de interés, preparar el plan de dirección del proyecto e iniciar otros procesos.
		7.2.3	Gestión de las interacciones	Gestión de las interacciones durante el proyecto.
		7.2.4	Gestión del cambio	Anticipación del cambio y su gestión a lo largo de todos los procesos.
		7.2.5	Cierre de procesos y proyecto	Cierre de los procesos y obtención de retroalimentación.
	7.3 Procesos relacionados con el alcance	7.3.2	Desarrollo de concepto	Definición de las especificaciones de lo que el producto del proyecto será.
		7.3.3	Desarrollo y control del alcance	Documentación de las características del producto del proyecto en términos mesurables para controlarlos.
		7.3.4	Definición de actividades	Identificación y documentación de las actividades y pasos requeridos para alcanzar los objetivos del proyecto.
		7.3.5	Control de actividades	Control el trabajo real llevado a cabo en el proyecto.
	7.4 Procesos relacionados con el calendario	7.4.2	Planeación de las dependencias entre actividades	Identificación de las interrelaciones y las interacciones lógicas así como las dependencias entre las actividades del proyecto.
		7.4.3	Estimación de la duración	Estimación de la duración de cada actividad en relación a condiciones específicas y los recursos requeridos.
		7.4.4	Desarrollo del cronograma	Interrelación de los objetivos de tiempo del proyecto, las dependencias de actividades y sus duraciones como marco de trabajo para desarrollar el cronograma general y detallado del proyecto.
		7.4.5	Control del cronograma	Control de la realización de las actividades del proyecto, para que se ajusten al plan propuesto o para tomar acciones corregir para corregir eventuales retrasos.
	7.5 Procesos relacionados con el costo	7.5.2	Estimación del costo	Desarrollo de las estimaciones de costo del proyecto.
		7.5.3	Presupuesto	Aprovechamiento de los resultados de la estimación de costos para producir el presupuesto del proyecto.
		7.5.4	Control de costos	Control de los costos y las desviaciones del presupuesto del proyecto.
	7.6 Procesos relacionados con la comunicación	7.6.2	Planeación de la comunicación	Planeación de los sistemas de información y comunicación del proyecto.

ISO 10006				
Cláusula	**Sub-cláusula**	**Sub-sub-cláusula**	**Proceso**	**Descripción del proceso**
		7.6.3	Gestión de la información	Colocación de la información necesaria disponible para la organización y grupos de interés del proyecto.
		7.6.4	Control de la comunicación	Control de la comunicación de acuerdo con el sistema planeado de comunicación.
	7.7 Procesos relacionados con el riesgo	7.7.2	Identificación del riesgo	Determinación de los riesgos del proyecto.
		7.7.3	Evaluación del riesgo	Evaluación de la probabilidad de ocurrencia de los eventos de riesgo y el impacto de éstos en el proyecto.
		7.7.4	Tratamiento del riesgo	Desarrollo de los planes de respuesta al riesgo.
		7.7.5	Control de riesgo	Implementación y actualización de los planes de riesgo.
	7.8 Procesos relacionados con compras	7.8.2	Planeación y control de las compras	Identificación y control de lo que se comprará y cuando se comprará.
		7.8.3	Documentar los requerimientos de compra	Recopilación de los requerimientos y condiciones comerciales de compra.
		7.8.4	Evaluación de proveedores	Evaluación y determinación de que proveedores pueden ser invitados a proveer productos y servicios.
		7.8.5	Contratación	Envío de invitaciones para hacer propuestas, evaluación de propuestas, negociación, preparación y cierre del contrato.
		7.8.6	Control de contratos	Asegurar que el desempeño de los proveedores cumple con los requerimientos contractuales.
8. Medición, análisis y mejora continua	8.1 Procesos relacionados con la mejora continua	8.1	Mejora continua	Establecimiento de guías en cuanto como la organización originadota y el proyecto deben aprender de los proyectos.
	8.2 Medición y análisis	8.2	Medición y análisis	Establecimiento de guías en la medición, recolección y validación de datos para la mejora continua.
	8.3 Mejora continua	8.3.1	Mejora continua en la organización originadora	Se deben tomar los pasos adecuados en la organización originadora para la mejora continua en los procesos del proyecto.
		8.3.2	Mejora continua en la organización del proyecto	La información que la organización del proyecto debe entregar a la organización originadora para permitir la mejora continua.

2.3.5. Capability Maturity Model Integration (CMMI)

Modelo para la mejora o evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. Fue desarrollado por el Instituto de Ingeniería del Software de la Universidad Carnegie Mellon (SEI), y publicado en su primera versión en enero de 2002.

El CMM - CMMI es un modelo de calidad del software que clasifica las empresas en niveles de madurez. Estos niveles sirven para conocer la madurez de los procesos que se realizan para producir software.

Los niveles de madures son cinco:

- **Inicial o Nivel 1:** Este es el nivel en donde están todas las empresas que no tienen procesos. Los presupuestos se disparan, no es posible entregar el proyecto en fechas, te tienes que quedar durante noches y fines de semana para terminar un proyecto. No hay control sobre el estado del proyecto, el desarrollo del proyecto es completamente opaco, no sabes lo que pasa en él.

- **Repetible o Nivel 2:** Quiere decir que el éxito de los resultados obtenidos se pueden repetir. La principal diferencia entre este nivel y el anterior es que el proyecto es gestionado y controlado durante el desarrollo del mismo. El desarrollo no es opaco y se puede saber el estado del proyecto en todo momento.

  Los procesos que hay que implantar para alcanzar este nivel son:

  - Gestión de requisitos
  - Planificación de proyectos
  - Seguimiento y control de proyectos
  - Gestión de proveedores
  - Aseguramiento de la calidad
  - Gestión de la configuración

- **Definido o Nivel 3:** Resumiéndolo mucho, este alcanzar este nivel significa que la forma de desarrollar proyectos (gestión e ingeniería) esta definida, por definida quiere decir que esta establecida, documentada y que existen métricas (obtención de datos objetivos) para la consecución de objetivos concretos.

  Los procesos que hay que implantar para alcanzar este nivel son:

  - Desarrollo de requisitos
  - Solución técnica
  - Integración del producto
  - Verificación
  - Validación
  - Desarrollo y mejora de los procesos de la organización
  - Definición de los procesos de la organización
  - Planificación de la formación
  - Gestión de riesgos
  - Análisis y resolución de toma de decisiones

La mayoría de las empresas que llegan al nivel 3 paran aquí, ya que es un nivel que proporciona muchos beneficios y no ven la necesidad de ir más allá porque tienen cubiertas la mayoría de sus necesidades.

- **Cuantitativamente Gestionado o Nivel 4:** Los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y la organización. Se usan métricas para gestionar la organización.

  Los procesos que hay que implantar para alcanzar este nivel son:

  - Gestión cuantitativa de proyectos
  - Mejora de los procesos de la organización

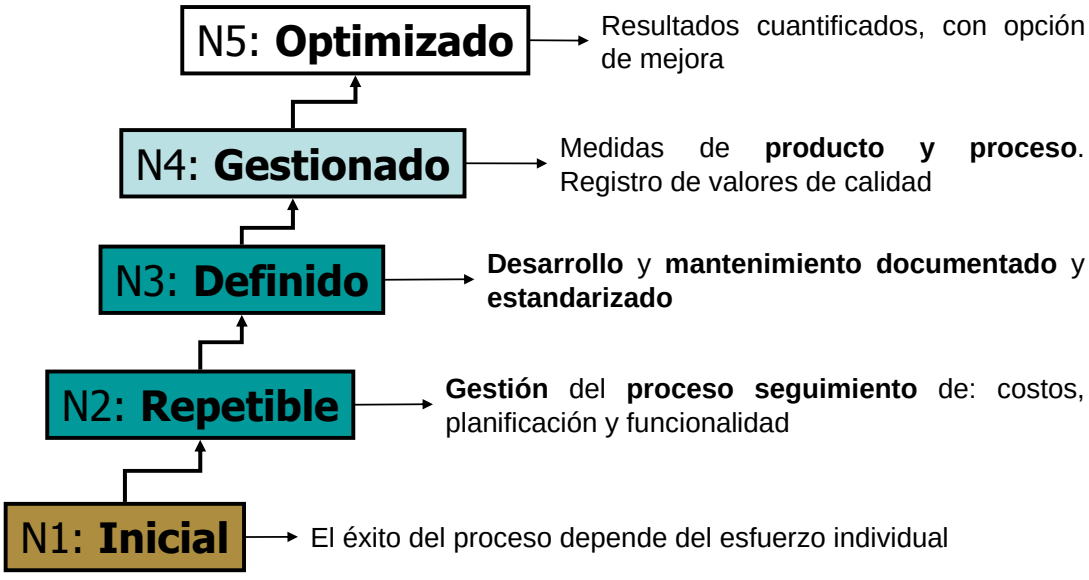
- **Optimizado o Nivel 5:** Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica.

  Los procesos que hay que implantar para alcanzar este nivel son:

  - Innovación organizacional
  - Análisis y resolución de las causas

  Normalmente las empresas que intentan alcanzar los niveles 4 y 5 lo realizan simultáneamente ya que están muy relacionados.

A continuación se muestra el modelo escalonado:



CAPÍTULO 3. PLANTEAMIENTO DEL PROBLEMA

3.1. INTRODUCCIÓN

Dentro de las áreas de Tecnología de la Información existen metodologías de desarrollo de sistemas, las cuales contiene un proceso de pruebas de aceptación del sistema para los sistemas que están en desarrollo como para los mantenimientos de aplicaciones en producción. Sin embargo, por diversas razones, se han enfrentado a problemas de diferentes naturalezas que repercuten en ineficiencias y retrasos en la entrega de productos, adicionalmente se encuentran errores de funcionalidad en los ambientes productivos que no se detectaron en la fase de pruebas.

Para cualquier metodología, la fase de pruebas de software es la actividad más visible en el proceso de aseguramiento de la calidad en software (SQA), la cual quizá sea la parte más crítica en un proyecto de desarrollo. No se trata de inspeccionar rápidamente los menús de una aplicación, no se trata de detectar algún defecto, se trata de asegurarse que el producto satisface las necesidades del cliente anticipándose a la aparición de algún error, éste es el objetivo principal de un proyecto de pruebas de software: documentar el error, localizarlo, buscar sus causas y repararlo. "Las pruebas a un programa puede revelar la presencia de algún defecto. Pero nunca su ausencia", una prueba con éxito será la que encuentre algún error antes de que lo haga el usuario final.

Buscando la optimización de este esfuerzo, las áreas de TI se han preocupado en la calidad de sus desarrollos de software para cumplir con las expectativas de las áreas usuarias en los requerimientos que soliciten y estos verse convertidos en productos de software con calidad, para ello necesita procesos formalmente definidos.

Algunos principales problemas que existen se listan a continuación:

- Estimación de tiempos basados en las tareas de desarrollo.
- Disminución de utilidades por errores en la planificación de los proyectos.
- No existe capacitación en pruebas de software.
- Los desarrolladores no realizan las pruebas unitarias.
- Desenfoque en las pruebas por realizar.
- Se realizan pruebas sin las herramientas apropiadas.
- Los retrasos que existen en la fase de desarrollo se mitigan con el tiempo asignado para ejecutar las pruebas.
- No existe una cultura enfocada a la calidad.
- Escasez de personal con formación y experiencia en nuevos métodos, normas y herramientas.
- Existen liberaciones de productos inmaduros, lo que trae como consecuencia una labor penosa y costosa de soporte de usuarios insatisfechos, además del gasto en el mecanismo para proveer a estos usuarios de una nueva versión con los problemas resueltos.
- La insatisfacción del cliente trae como consecuencia pérdida de imagen del área, así como una baja de la credibilidad en la misma.
- En sistemas críticos la falla del sistema trae como consecuencia pérdidas monetarias, con repercusiones económicas en los estados financieros de la compañía.

3.2. REQUERIMIENTOS GENERALES Y PARTICULARES (ALCANCE)

Las áreas de aseguramiento de calidad tienen la firme idea de fortalecer la calidad en los proyectos, por lo que es necesario redefinir un proceso de pruebas con las mejores practicas existentes en el mercado.

Por lo que en esta tesis, se han establecido objetivos para fortalecer los procesos de pruebas de una organización. Los requerimientos se pueden acotar en los siguientes puntos:

- Definir el proceso formal para la administración de pruebas de software.
- Incrementar la eficiencia administrativa, operativa y tecnológica en la aplicación de pruebas al software.
- Mejorar la capacidad instalada y disminuir los costos de operación.
- Aseguramiento de la adherencia y el cumplimiento de los procesos y procedimientos organizacionales.
- Aseguramiento de la calidad de los productos de trabajo generados en un proyecto.
- Generar planes de calidad para cada proyecto.
- Generar planes de calidad para áreas específicas dentro de una organización.
- Capacitación de procesos, prácticas y formatos.
- Seguimiento al cierre de los hallazgos detectados en cada auditoria.
- Recolectar mejoras a los procesos actuales.
- Capacitación y asesorías de los procesos actuales y al momento de que surja algún cambio o mejora de los mismos.
- Ajustes de los procesos organizacionales a un proyecto.
- Recopilación y administración de candidaturas de mejoras a los procesos.
- Evaluación del staff de pruebas y del servicio prestado (encuestas de satisfacción).
- Visibilidad (generación de métricas).
- Mejorar el tiempo de respuesta de atención de los requerimientos de prueba de los usuarios.
- Mejorar la calidad de los servicios de soporte y mantenimiento.
- Reducir los plazos para la entrega de los servicios.
- Definir con claridad el perfil de los ingenieros de pruebas de software que contribuyan a disminuir los riesgos técnicos.
- Fortalecer las habilidades y capacidades técnicas de los ingenieros de pruebas de software.
- Administrar las capacidades de los ingenieros de pruebas de software mediante un pool de servicios.
- Uniformar o estandarizar los instrumentos automatizados para los patrones de prueba de software.
- Definir los estándares para el diseño de sets de pruebas.
- Definir los diferentes tipos de pruebas: unitarias, integrales, de concurrencia, de esfuerzo, modulares, de estrés, de reglas de negocio, de base de datos y cualquiera otra conocida o por conocer.

Al ser esta una actividad de mucha visibilidad y trascendencia dentro de las organizaciones, todas las consideraciones realizadas para el planteamiento de los recursos tanto humanos como técnicos, se han enfocado a minimizar los riesgos y tener en consideración todos los factores que pudieran llegar a afectar en la ejecución y obtención de los resultados deseados.

3.3. RECOPILACIÓN Y ANÁLISIS DE LA INFORMACIÓN

La razón principal de esta tesis fue analizar la situación actual de las áreas de Tecnología de la Información en México en referencia a sus prácticas y procesos de prueba para detectar las áreas de oportunidad existentes, así como sus fortalezas.

3.3.1. Estrategia

El diagnóstico está completamente enfocado a identificar si una organización cuenta con procesos y prácticas sobre la disciplina de pruebas y si es así, conocer qué tan "buena" es ésta, identificando mecanismos de control básicos, para de esta forma visualizar el nivel de madurez de los procesos y prácticas existentes en la organización.

Para realizar lo anterior se siguieron las siguientes fases:

Actividades generales

- Realizar reuniones con cada uno de los involucrados, identificando si existe o no un proceso formal de desarrollo. Si existe un proceso formal revisar el proceso con el fin de identificar las prácticas asociadas en las diferentes disciplinas.
- Recabar la información existente del proceso actual para realizar un análisis y tener una visión más clara de lo que hace falta en él.
- Documentar la información recabada para poder contar con un registro de ella.
- Analizar la información para la detección de problemas, deficiencias así como las oportunidades existentes.
- Generar reportes de la situación actual, así como estrategias para su mejora.

De acuerdo al modelo anterior se obtuvieron los resultados que se describen en los siguientes puntos del documento, se detallan los resultados encontrados en la disciplina de pruebas y al final se muestran las conclusiones globales.

3.3.2. Modelo general

Normalmente una área de Aseguramiento de Calidad esta encargada de dar seguimiento a las pruebas e incidencias reportadas por parte de los usuarios, enfocándose principalmente a esta última en la cual se envían reportes diarios sobre el estatus de los errores.

El grupo de trabajo asignado para llevar a cabo la tarea de establecer y asegurar las pruebas a las aplicaciones que se desarrollan así como a los mantenimientos y errores productivos normalmente es pequeño.

La labor de crear y ejecutar pruebas es realizada principalmente por usuarios finales quienes muchas veces no cuentan con el tiempo necesario para diseñar o ejecutar pruebas que cubran en su totalidad el funcionamiento de las aplicaciones, además de no estar asignados 100% a la realización de esta tarea.

Comúnmente, la experiencia es una guía para establecer métodos de trabajo enfocados a ejecutar pruebas integrales (E2E) y pruebas de aceptación (UAT), así como para crear algunos artefactos (matrices y casos de prueba), aunque esta practica sólo se lleva en algunos proyectos, esta es una de las razones por las cuales se ha identificado como prioritario contar con un proceso formal de pruebas para garantizar la calidad del software que se libera a producción.

3.3.3. Análisis por disciplinas

EQUIPOS DE PRUEBAS

Las áreas de Tecnología de Información normalmente, cuentan con equipos de pruebas formados por:

- Subdirector de Aseguramiento de Calidad
- Coordinadores
- Ejecutores de pruebas (probadores o testers)

Los coordinadores de prueba se enfocan al seguimiento de la corrección y validación de errores y la comunicación entre los proveedores y usuarios.

La asesoría que se brinda en los proyectos sobre los aspectos que se deberían de probar en las aplicaciones no es una práctica común.

A pesar de contar con el equipo de pruebas, en ocasiones no se cuenta con una metodología implementada que permita llevar acabo una buena labor como equipo de pruebas de software.

Por lo anterior, se observa una gran necesidad de implementación de un proceso formal de pruebas así como la capacitación formal sobre pruebas de software.

Se llevó a cabo una comparativa de lo que contempla el modelo TMM con respecto a los estándares para pruebas de software contra lo que se contempla los modelos V, W y X.

De lo anterior, se observó que era necesario implementar un proceso que este estructurado para llevar un correcto control de lo que será probado (análisis, planeación y parte del diseño), lo probado (más diseño y ejecución) y los resultados obtenidos (control y seguimiento de errores), además de contener una fase de documentación, estabilización y

validación del producto final (pruebas de aceptación). El Proceso debe proponer aplicar las pruebas de forma iterativa e incremental; conforme se va realizando el avance del proyecto, se debe intensificar la realización de cada una de las pruebas asegurando una máxima cobertura.

Así mismo, este proceso debe contemplar una fase de administración de pruebas y evaluación de pruebas; la primera se refiere a organizar, manipular y controlar los artefactos generados dentro de un proyecto tales como requerimientos, planes, documentación, scripts y resultados de pruebas (por mencionar algunos activos del testware). En el caso de la fase de evaluación de pruebas se debe enfocar en analizar las mejores y peores prácticas para tener una mejora continua. En este caso esta fase se aplicará al terminar cada ciclo de pruebas para replicar lo que se esta haciendo bien, eliminar lo que no se esta haciendo bien y seguir robusteciendo el diseño y ejecución de pruebas (esta fase también la podemos llamar post-mortem).

A continuación se muestra una tabla con los artefactos que se deben de manejar para la correcta documentación de un proceso de pruebas de software según el TMM.

Artefactos	Proceso de pruebas
Plan de pruebas	✓
Especificación del diseño de pruebas	✓
Especificación de casos de prueba	✓
Especificación de procedimientos de pruebas	✓
Bitácora de pruebas	✓
Reporte de incidencias de prueba	✓
Reporte de pruebas	✓

En la siguiente tabla se muestran los atributos que todo proceso de pruebas y staff maduros deben contener de acuerdo a lo señalado por TMM

Atributo	Proceso de pruebas
Cuenta con un conjunto de políticas o prácticas de prueba	✓
Proceso de planeación de pruebas	✓
Ciclo de vida de las pruebas	✓
Staff de pruebas	✓
Staff para la mejora del proceso de pruebas	✓
Conjunto de métricas de prueba	✓
Administración y seguimiento	✓

3.4. IDENTIFICACIÓN DEL PROBLEMA

Derivado de la información recabada anteriormente para identificar la problemática existente en los procesos de pruebas, se han detectado los siguientes aspectos:

ÁREAS DE OPORTUNIDAD

- No existe ni un proceso formal de pruebas, ni un conjunto de prácticas que se realicen de manera estándar en todos los desarrollos, lo que permitirá que la adopción de prácticas ágiles y puntuales sea de una forma rápida.
- No se cuenta con el conocimiento deseado sobre lo que son las pruebas de software y cómo aplicarlas.
- Actualmente las áreas de calidad no se involucra desde el análisis de nuevos requerimientos.
- No existe una planeación adecuada del tiempo que requieren las pruebas del sistema.
- Durante la definición de un sistema no se especifican los requerimientos no funcionales a probar, y al no tener perfectamente identificados los requerimientos de calidad asociados como performance o concurrencia puede impactar la operación de la aplicación cuando tenga un uso real en producción.
- La asignación de usuarios finales como staff de prueba no asegura que los sistemas sean verificados en su totalidad, lo que incrementa el riesgo de encontrar errores durante la operación de los sistemas en un ambiente de producción.
- En ocasiones hay integrantes del equipo de desarrollo dando asesoría a los usuarios durante la ejecución de pruebas.
- Un punto importante es la migración de la aplicación entre ambientes de integración (E2E) y de aceptación (UAT). Al cambiar de ambientes el sistema no se lleva a cabo una verificación de la funcionalidad, lo que representa que el usuario tenga que volver a hacer la misma validación que en el ambiente previo.
- La confianza que tienen los usuarios en las aplicaciones que utilizan regularmente no es del 100%, lo que genera incertidumbre sobre la calidad de éstos. No tienen la seguridad de que la forma de operar en producción será la que esperan.
- Las pruebas unitarias ejecutadas por el equipo de desarrollo antes de entregar una aplicación son de forma manual y no se genera ninguna evidencia de éstas. En algunos casos han tenido el apoyo de elementos que les ayudan a saber si las operaciones de los sistemas son correctas, aunque no es siempre.
- No todas las personas involucradas en los proyectos conocen la documentación que hoy en día se utiliza para la creación de matrices y casos de prueba, por lo que muchas veces se reinventan.
- Al día de hoy pueden conocer el estatus de las pruebas basados en la cantidad de errores y sus estatus pero no pueden saber con certeza que cobertura de la funcionalidad ha sido probada.

FORTALEZAS

- El esfuerzo de implementar procesos y prácticas de desarrollo y pruebas es respaldado a nivel directivo y forma parte de los principales proyectos del área de Tecnología de la Información.
- Existe cultura organizacional de pruebas, donde cada una de las áreas tienen conciencia de la importancia de éstas dentro del desarrollo del software, lo cual permite que sea mucho más sencilla la implementación de un proceso estándar de pruebas.
- En ocasiones, se cuentan con herramientas de control y seguimiento de errores.
- Durante la ejecución de pruebas de aceptación, se documenta la evidencia de pruebas que se realizaron a una aplicación, los cuales son validados por las diferentes áreas.

RECOMENDACIONES

De acuerdo a la situación general, se recomienda una aproximación que incluya enfocar los esfuerzos iniciales en establecer formalmente el área de pruebas en la organización, la implementación de procesos de control de la calidad son uno de los elementos críticos en las empresas, así mismo de acuerdo a la organización actual del equipo de calidad, la formalización del área tendría resultados a muy corto plazo y que serían observables a nivel de toda la organización.

Se recomienda involucrar al área de pruebas desde las actividades asociadas con el análisis de requerimientos hasta los procesos de liberación y transición de un sistema a producción, así como su participación en mantenimientos a aplicaciones productivas y corrección de errores encontrados en este ambiente.

Así mismo, una de las mayores recomendaciones es el uso de equipos de prueba para el diseño y ejecución de pruebas antes de involucrar a un usuario final, esto daría como resultado que las aplicaciones tuvieran una verificación mas robusta en funcionalidad y permitir que el tiempo asignado de un usuario sea exclusivamente para la validación de operaciones de negocio.

CAPÍTULO 4. PROPUESTA DE SOLUCIÓN

4.1. INTRODUCCIÓN

En este capítulo se describe el conjunto de acciones que se llevarán acabo durante el diseño del proceso de pruebas a documentar, basado en las conclusiones obtenidas durante la etapa de análisis realizado, donde se detectaron áreas de oportunidad y fortalezas que pueden apoyar en el desarrollo de cualquier proyecto.

Se definirán los siguientes puntos:

- Actividades
- Recursos necesarios
- Personas responsables
- Marco de tiempo

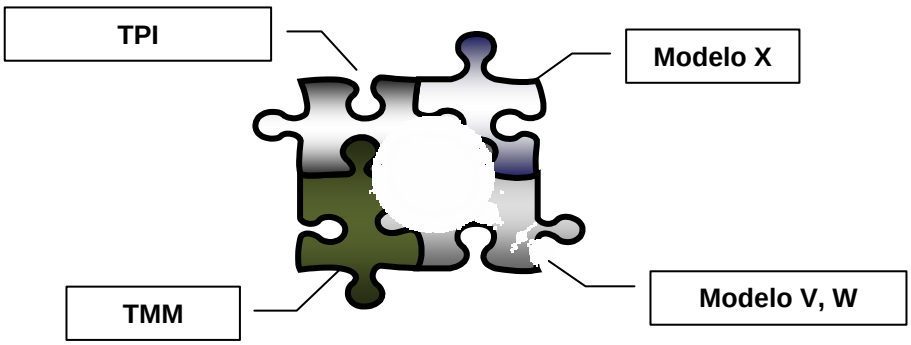
Se enlistarán las estrategias, las soluciones que se llevarán a cabo, así como la forma en la cual se realizarán.

4.2. PLAN DE ACCIÓN

4.2.1. Solución propuesta

De acuerdo a lo que se detectó durante la fase de "recopilación, análisis y evaluación" se propone la siguiente solución:

- Diseño de una metodología de pruebas basada en las siguientes metodologías así como en las mejores prácticas:
  - TMM → Modelo de Madurez de Pruebas (Testability Maturity Model).
  - TPI → Modelo de Mejora del Proceso de Pruebas (Test Process Improvement).
  - Modelos V, W, X



- Definición de funciones y responsabilidades (asignación de roles).
- Capacitación del personal del equipo de pruebas dentro de las organizaciones de TI.
- Ajustar y personalizar artefactos (templates, formatos, guías) de acuerdo a las

necesidades del área de Aseguramiento de Calidad y a la integración a las fases de cualquier metodología de desarrollo de sistemas.

- Proposición de un esquema de trabajo a utilizar para almacenamiento y como repositorio de la documentación generada dentro del área de pruebas.
- Realizar un plan de trabajo detallado con cada una de las actividades a realizar, definiendo los tiempos, recursos y responsables de cada una de ellas.

4.2.2. Acciones propuestas

A continuación se listan las acciones que se debe seguir para implantar una metodología de pruebas, así como los tiempos sugeridos y el responsable de ejecutar las acciones propuestas:

Implantación de la metodología			
Acción	Recursos necesarios	Responsable	Tiempo
Generación del plan de acción	N/A	Consultor experto en pruebas	24 horas
Definición gráfica de la forma de trabajo. Definición de la interacción del área de pruebas con el área de desarrollo	N/A	Consultor experto en pruebas	9 horas
Generar sitio de colaboración	Apoyo del área de Infraestructura. Servidor y accesos necesarios.	Consultor experto en pruebas y apoyo del personal del área de infraestructura	22 horas
Taller de pruebas	Sala de capacitación oficinas de la organización	Consultor experto en pruebas	40 horas
Definición de sugerencias de los roles	N/A	Consultor experto en pruebas	15 horas
Adecuación del proceso de pruebas a las necesidades de la empresa	N/A	Consultor experto en pruebas	12 horas
Adecuación y diseño de guías	N/A	Consultor experto en pruebas	11 horas
Adecuación y diseño de plantillas	Apoyo del equipo de Aseguramiento de Calidad IT	Consultor experto en pruebas	11 horas
Entrega y validación de resultados	N/A	Consultor experto en pruebas y subdirector de aseguramiento de calidad IT	3 horas
Revisión del proceso de pruebas	Sala de juntas	Consultor experto en pruebas y subdirector de aseguramiento de calidad IT	3 horas
Total:			150 horas

Asesoría dentro del proyecto piloto			
Acción	Recursos necesarios	Responsable	Tiempo

Durante todo el Proyecto			
Acción	Recursos necesarios	Responsable	Tiempo
Administración de pruebas	N/A	Consultor experto en pruebas	N/A

4.3. DESCRIPCIÓN DE LAS ACCIONES PROPUESTAS

4.3.1. Definición de la forma de trabajo

Para poder implementar el nuevo proceso de pruebas se considerará un conjunto de mejores prácticas y técnicas de desarrollo de software, teniendo como propósito mejorar el nivel de calidad del software desarrollado e incrementar el control del proceso de desarrollo de sistemas.

El principal enfoque es, ser pragmático y enfocado en la gente, de tal forma que al ser un conjunto de mejores prácticas permitan seleccionar las herramientas adecuadas para montar un proceso de pruebas ágil que apoye a los desarrolladores a mantener un nivel consistente de calidad, colaboración y control del código.

Estas mejores prácticas han demostrado sus beneficios y han sido adoptadas por la industria de software a nivel mundial para mitigar problemas que típicamente aparecen durante el proceso de desarrollo de software y que ponen en riesgo la ejecución de los proyectos. Entre las prácticas que incorporaremos son:

- Sitio de colaboración.
- Seguimiento de incidentes.
- Elaboración de pruebas unitarias, funcionales, de integración y de aceptación.
- Automatización de tareas.
- Integración continúa.
- Creación de casos y matrices de prueba.
- Revisión de estándares de codificación.
- Análisis estático de código.

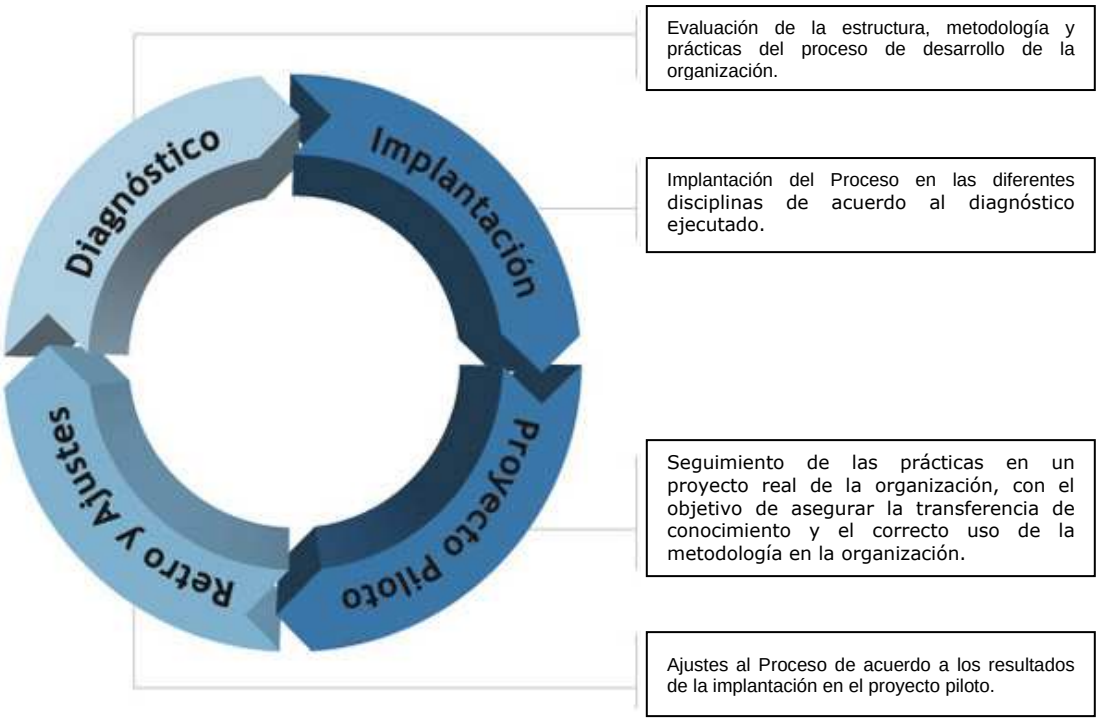
Estas prácticas pueden ser fácilmente adaptadas a las necesidades específicas del proyecto a ejecutar y del equipo humano, así como adecuarse a las prácticas de desarrollo ya establecidas dentro de la organización.

Para llevar a cabo la implementación del nuevo proceso de pruebas se requerirá del apoyo y validación del personal de las áreas de Aseguramiento de Calidad IT.

Trabajando con los siguientes escenarios:

- Nuevo desarrollo de proyectos
- Mantenimientos a sistemas actuales
- Corrección de errores productivos

A continuación se muestra las actividades que se llevaran a cabo para la implementación del nuevo proceso de pruebas de acuerdo al siguiente



Definición de la Interacción del área de pruebas con el área de desarrollo: De acuerdo a las actividades definidas, se definirá de manera detallada, cuál es la interacción que el área de pruebas tendrá con el área de desarrollo. Así como cuáles serán las actividades, responsables y los entregables de cada una de las áreas.

4.3.2. Definición de sugerencias de los roles para el staff de pruebas

Se definirán sugerencias de roles a desempeñar dentro de un equipo de pruebas, así como las actividades y responsabilidades a realizar por parte de cada uno de ellos.

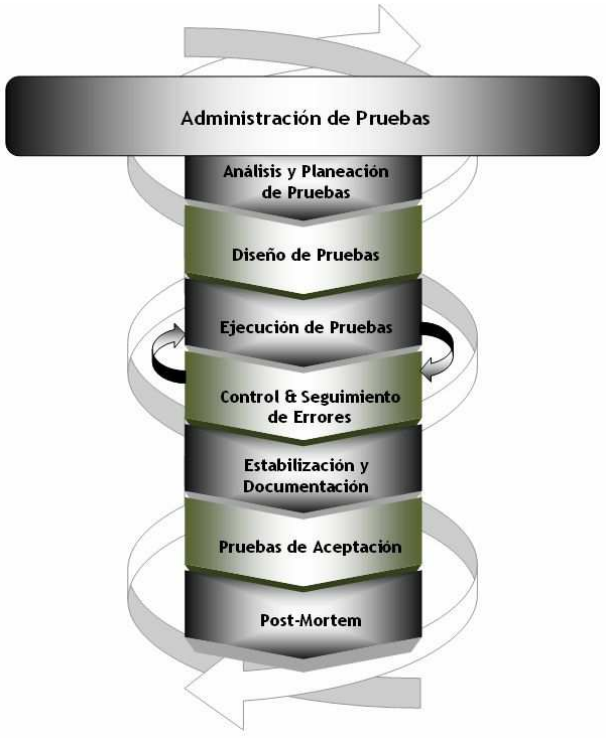
4.3.3. Adecuación del proceso de pruebas a las necesidades del área de Aseguramiento de Calidad

Se utilizará el nuevo proceso de pruebas, el cual será integrado dentro de las metodologías de desarrollo que utilicen las organizaciones, considerando los siguientes procesos:

- Administración de pruebas
- Análisis y planeación de pruebas
- Diseño de pruebas
- Ejecución de pruebas
- Control y seguimiento de errores
- Estabilización y documentación
- Pruebas de aceptación
- Post mortem

Dentro de cada una de estas fases se realizan diferentes actividades y existen entregables en algunas de ellas. Este proceso se analizará para adecuarse a las necesidades propias de la organización, para con esto garantizar que el proceso de pruebas funciona correcta y eficientemente para la organización.

De forma gráfica el proceso se muestra a continuación:



Descripción: Este proceso que esta estructurado para llevar un correcto control de lo que será probado (análisis, planeación y parte de diseño), lo probado (más diseño y ejecución) y los resultados obtenidos (control y seguimiento de errores), además de contener una fase de documentación, estabilización y validación del producto final (pruebas de aceptación); concluyendo con una fase de evaluación (análisis de las mejores y peores prácticas de cada ciclo de pruebas, para tener una mejora continua).

Todo lo anterior organizando, manipulando y controlando los artefactos generados dentro de un proyecto (administración de pruebas).

Algo muy interesante de este proceso es que al tener todo el testware documentado y con la creación de un correcto diseño de pruebas (procedimientos, mapas, casos y scripts de pruebas) provee todo lo necesario para su reutilización.

Características principales del proceso de pruebas:

- ✓ Provee todo lo necesario para su *reutilización*.
- ✓ Las pruebas son aplicadas de manera *iterativa incremental*.
- ✓ Cuenta con *guías y formatos* de trabajo para saber ¿Qué hacer? y ¿Cómo hacerlo?
- ✓ Proceso *muy práctico*.
- ✓ Cuenta con una *extensión* para soportar mejores prácticas y otras disciplinas de pruebas.
- ✓ Propone empezar a *probar desde las fases tempranas del desarrollo*.

4.3.4. Adecuación y diseño de guías

Se adecuarán y en su caso, se realizarán las guías para cada una de las fases del proceso. En estas guías se explicará de forma detallada la manera en la cual se trabaja en cada fase, los pasos a seguir, los entregables a generar y como utilizarlos. Dentro de las guías se contemplan:

- Guía para la administración de pruebas
- Guía de análisis y planeación de pruebas
- Guía del diseño de pruebas
- Guía de la ejecución de pruebas
- Guía para el control y seguimiento de errores
- Guía sobre la fase de estabilización y documentación
- Guía para pruebas de aceptación
- Guía sobre la fase del post mortem

Estas guías apoyarán al equipo de pruebas en la aplicación del proceso dentro de los proyectos que se lleven a cabo. Cabe mencionar que estas guías serán útiles para la capacitación de nuevos elementos del staff de pruebas.

4.3.5. Adecuación y diseño de plantillas

En algunas de las fases será necesario generar entregables, por lo que se adecuarán y/o generarán plantillas teniendo así, una base de los puntos a considerar así como una breve descripción de cada punto o un ejemplo de lo que debe contener cada una de ellas.

Las plantillas que se adecuarán serán de acuerdo a las mejores prácticas de pruebas.

4.3.6. Revisión del proceso de Pruebas

Una vez definido, diseñado e integrado el nuevo proceso de pruebas a la metodología de desarrollo de sistemas se debe contemplar presentarlo a todas las áreas involucradas en el desarrollo de sistemas para dar a conocer la nueva forma de trabajo para el área de Aseguramiento de Calidad de IT y su interacción para el asegurar la calidad en las diferentes aplicaciones.

4.4. PLAN DE TRABAJO GENERAL

Plan de trabajo general. Es el calendario a alto nivel, sin el detalle de la realización del proyecto, en el cual se tienen las siguientes fases para lograr los objetivos y alcances del proyecto.

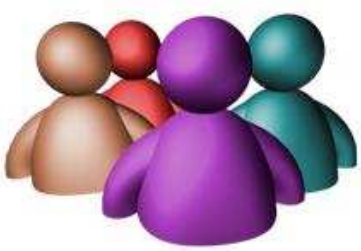
Fase / Semanas	1					2					3					4					5					6				
1. Diagnóstico																														
1.1. Kick Off																														
1.2. Inducción y planeación																														
1.3. Recopilación de información																														
1.4. Análisis de resultados																														
1.5. Documentación y validación																														
1.6. Aceptación del reporte de diagnóstico																														
2. Implantación																														
2.1. Planeación																														
2.2. Adecuación de mejores prácticas																														
3. Entrenamiento																														
4. Revisión del proceso de pruebas																														
4.1. Revisión																														
4.2. Cierre																														

CAPÍTULO 5. IMPLEMENTACIÓN DEL PROCESO DE PRUEBAS

5.1. INTRODUCCIÓN

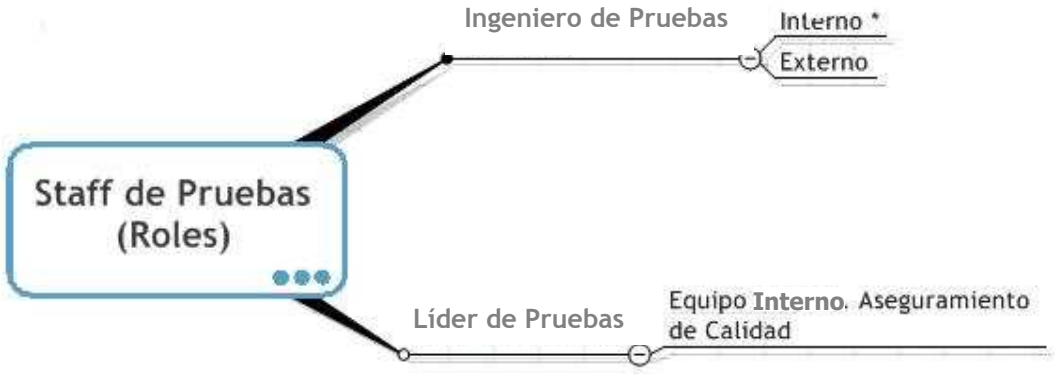
Dentro de este capítulo se describen los procesos generados, así como sus entregables por cada fase y el acoplamiento en la metodología de desarrollo de sistemas.

5.2. DESCRIPCIÓN DE ROLES



Un buen profesional de pruebas debe tener la actitud de que "**siempre habrá algo que probar**", debe tener la habilidad de tener el punto de vista del cliente, un gran deseo de calidad y mucha atención por el detalle. También debe tener tacto y diplomacia en el trato con los clientes y con su mismo equipo. La experiencia le va dando al ingeniero de pruebas la capacidad de identificar puntos clave dentro del proceso de pruebas cuando éste se aplica justo a tiempo.

5.2.1. Roles del área de pruebas



HABILIDADES GENERALES

- Identificación con el usuario, habilidad para la organización y comunicación, y actitud para la solución de problemas.
- Habilidad para generar documentación de pruebas.
- Creatividad y alto grado de motivación.
- Interés por nuevos desafíos.
- Habilidad para trabajar en equipo y organizar su trabajo de manera independiente.
- Tomar en cuenta sugerencias para la mejora en su trabajo.
- Tener la habilidad de evadir presiones de todo tipo.
- Mezcla de conocimientos técnicos y de pruebas.

- Conocimiento en desarrollo de software: Dependiendo de la complejidad del proyecto, el ingeniero de pruebas debe tener los suficientes conocimientos técnicos para tratar un problema tecnológico.
- Habilidad verbal: El ingeniero de pruebas es encargado de realizar planes que deben de ser comprendidos por el equipo de trabajo, debe de ser capaz de expresar las metas planteadas en el proyecto. Su trabajo depende del trabajo de muchas personas, la buena comunicación con el equipo de desarrollo es importante, para tener a todo el equipo integrado con los errores encontrados.
- Organización: Es imposible probar todo, por lo tanto el ingeniero de pruebas debe de ser lo suficientemente organizado para ejecutar las pruebas de acuerdo a prioridades y debe de poder llevar un control de las versiones que se han probado. No siempre se cuenta con el tiempo y con las herramientas para desarrollar el plan de trabajo, por lo que una de las principales metas es ser eficiente para llevar a cabo cada una de las actividades.
- Experiencia en pruebas: Ira acumulando con la participación de proyectos o bien con la disposición de trabajar en versiones betas de diferentes productos para diferentes empresas.
- Actitud: Creatividad destructiva, para sobrepasar los límites de un programa. No importa que el desarrollador diga que nada cambio siempre hay algo que no funciona y nuestra tarea es encontrarlo.
- Y deberá siempre de tratar de realizar diferentes tipos de pruebas que no estén estipuladas como básicas ya que eso le permitirá encontrar diferentes tipos de errores.

Ingeniero de pruebas (STE - Software Test Engineer)

1. Conocimientos:

   - Ingeniería en Sistemas Computacionales o carrera afín.
   - Bases matemáticas.
   - Conocimiento en bases de datos.
   - El candidato ideal debe tener conocimientos básicos en Ingeniería de Software.
   - Inglés 80%.

2. Capacitación requerida:

   - Introducción a las pruebas software.
   - Conocimiento en modelos de pruebas.
   - Fundamentos de automatización de pruebas.
   - Diseño y ejecución de pruebas.
   - Control y seguimiento de errores.

3. Habilidades:

   - Capacidad de extraer la funcionalidad de un producto, documentar los resultados de esta extracción a manera de casos de prueba así como ejecutar los casos de prueba para verificar la correcta funcionalidad del producto.
   - Habilidad para el diseño y ejecución de pruebas.
   - Capacidad para controlar y dar seguimiento a los errores.

   Nota: Requiere que se le supervise y se le diga claramente que es lo que hay que hacer y en algunos casos el cómo.

4. Responsabilidades:
   - Planear, diseñar y aplicar las pruebas del proyecto al cual fue asignado.
   - Controlar y dar seguimiento a los errores del proyecto al cual fue asignado.
   - Documentar las pruebas del proyecto y sus resultados.
   - Reportar sus actividades, riesgos y situación del proyecto al líder de pruebas de software.
   - Garantizar la satisfacción de clientes y usuarios en cuanto al cumplimiento de sus requerimientos.
   - Mantenerse en contacto con el cliente para validar las pruebas a realizar.
   - Garantizar el nivel de calidad del software.

5. Experiencia: De 6 meses a 1 año (deseable)

Líder de pruebas de software (STL - Software Test Lead)

1. Conocimientos:

   - Ingeniería en Sistemas Computacionales o carrera afín.
   - Bases matemáticas.
   - Conocimiento de por lo menos una metodología y/o procesos de calidad y pruebas.
   - Manejo de bases de datos. Tener la suficiente experiencia que le permita automatizar pruebas (cargando tablas de la base de datos, importando, exportando, y seleccionando datos).
   - Uso de cualquier herramienta de automatización de pruebas (Compuware, Mercury Interactive, RSW, Segue, Rational Robot, Rational Test RealTime, etc.).
   - Manejo de herramientas para el control y seguimiento de errores.
   - Administración de equipos de trabajo.
   - Inglés 80%.

2. Capacitación requerida:

   - Administración de pruebas.
   - Administración de proyectos.
   - Administración de riesgos (Risk Based-Testing).
   - Introducción a las pruebas software
   - Conocimiento en modelos de pruebas.
   - Diseño de pruebas.
   - Control y seguimiento de errores.
   - Fundamentos de automatización de pruebas.

3. Habilidades:

   - Habilidad para desarrollar e implementar las adecuaciones necesarias al proceso estándar de pruebas de acuerdo al proyecto a probar.
   - Desarrollo y análisis de métricas.
   - Desarrollo de scripts de pruebas reutilizables.
   - Dirigir y coordinar a un equipo de ingenieros de prueba (equipos de 2 a 3 personas) siendo el responsable de la calidad de ciertas áreas y/o componentes de un sistema.
   - Capacidad para la correcta comunicación (cliente, equipos de desarrollo - corrección de errores, staff de pruebas, etc.).

- Habilidad para delegar y asignar actividades.
- Capacidad de entender el diseño técnico del sistema y de desarrollar especificaciones de software de acuerdo a los requerimientos.
- Habilidad para automatizar.
- Habilidades analíticas sólidas.
- Atención por el detalle y flexibilidad (mente abierta).

4. Responsabilidades:

   - Liderar un equipo en la planeación, diseñó y aplicación de las pruebas del proyecto al cual fue asignado, así como conducir al equipo de pruebas al éxito.
   - Controlar y dar seguimiento a los errores del proyecto al cual fue asignado.
   - Crear y mantener los planes y métodos de pruebas con exactitud e integridad.
   - Mantener el plan de trabajo, reportes de errores, resultados de pruebas y reportes de estatus (documentación de las pruebas del proyecto y sus resultados).
   - Dirigir y coordinar a un equipo de ingenieros de prueba (equipos de 2 a 3 personas).
   - Reportar sus actividades, riesgos y situación del proyecto al gerente de pruebas de software.
   - Instruir a los ingenieros de pruebas de software acerca de la manera correcta y apropiada de poner en marcha procedimientos de pruebas.
   - Garantizar la satisfacción de clientes y usuarios en cuanto al cumplimiento de sus requerimientos.
   - Mantenerse en contacto con el cliente para validar las pruebas a realizar.
   - Mantenerse en contacto con el BSA, BA y/o proveedores según sea el caso para validar las pruebas a realizar.
   - Garantizar el nivel de calidad del software.

5. Experiencia: De 3 a 5 años de experiencia en el ramo.

   - Experiencia en aseguramiento de calidad y pruebas de software.
   - Experiencia en el manejo de equipos de trabajo (equipos pequeños y medianos). Así mismo experiencia en el asesoramiento en procesos de pruebas.
   - Experiencia en diferentes métodos de pruebas (automatización de pruebas, priorización de pruebas, leguajes de codificación de pruebas).
   - Diseño, ejecución y automatización de pruebas.
   - Experiencia en el uso de herramientas de control y seguimiento de errores.
   - Experiencia de en la creación de estrategias de pruebas.
   - Experiencia en por lo menos un proyecto como responsable de un equipo de pruebas.
   - Participación en por lo menos dos proyectos de misión crítica.
   - Participación en las pruebas de cinco ó más proyectos completos.

5.2.2. Actividades relacionadas con otras áreas

Este anexo describe la relación y responsabilidades que tienen otras áreas involucradas en el desarrollo de sistemas con el área de pruebas y con los procesos.

Analista de Sistemas de Negocio (BSA - Business System Analyst)

1. Responsabilidades:

- Notificar al área de pruebas cuando se acepten los requerimientos para un nuevo proyecto.
- Notificar al área de pruebas cuando se acepte una solicitud de cambio a un sistema actual.
- Requerir apoyo al área de Aseguramiento de Calidad para el detalle de los documentos de requerimientos de negocio, de esta forma se incorporará al equipo de pruebas desde el inicio del proyecto.
- Proveer de toda la información necesaria sobre el proyecto derivada del análisis y diseño.
- Involucrar al líder de pruebas asignado en cualquier actividad referente al proyecto.
- Dar seguimiento a la corrección de errores que el equipo de pruebas registre en la herramienta de seguimientos de defectos.
- Apoyar y asesorar al equipo de pruebas con cualquier problema técnico que se pudiera presentar.
- Recibir Vo.Bo. del líder de pruebas asignado cuando la aplicación ya se encuentre estable en el ambiente de pruebas integrales.

Analista de Negocio (BA - Business Analyst)

1. Responsabilidades:

- Aprobar la documentación generada del proceso.
- Dar seguimiento a los defectos registrados en la herramienta de seguimientos de defectos.
- Apoyar a los usuarios en la definición de requerimientos a sistemas.
- Involucrar al líder de pruebas asignado al proyecto en la documentación de requerimientos de negocio.
- Apoyo al equipo de pruebas en la aclaración de dudas sobre requerimientos de negocio.
- Seguimiento a las actividades planteadas en el análisis y planeación de pruebas.
- Coordinar en conjunto con el equipo de pruebas la logística para pruebas de aceptación con usuarios.
- Ejecución de pruebas de aceptación.
- Notificar al equipo de pruebas sobre cambios en los requerimientos.

Administrador de contratos (CA - Contract Administrator)

1. Responsabilidades:

   - Proveer los equipos necesarios (hardware y software) para la ejecución de pruebas.
   - Involucrarse en la definición de requerimientos no funcionales de la aplicación.
   - Recibir el Vo.Bo. de equipo de pruebas para migrar la versión del sistema al ambiente de preproducción.
   - Recibir el Vo.Bo. de los usuarios para migrar la versión del sistema al ambiente de producción.
   - Mantener homologados los ambientes de prueba para que sean lo más parecido a producción.

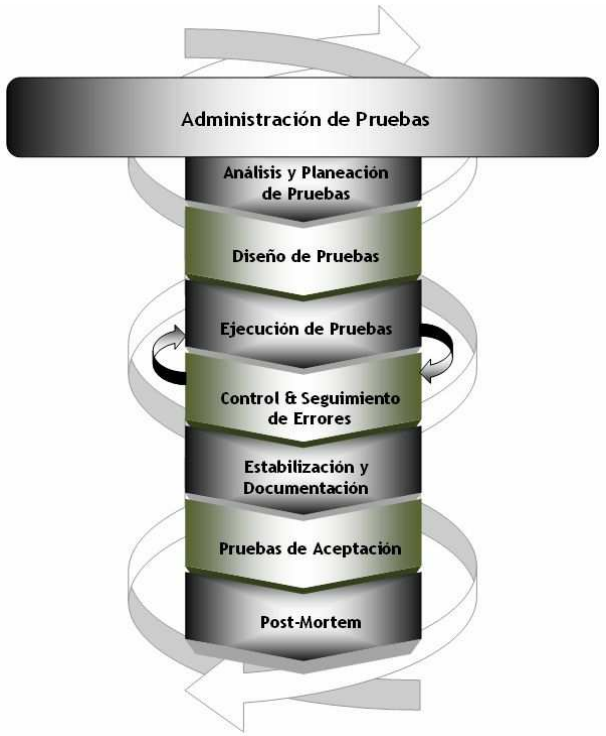
Usuario final (End user)

1. Responsabilidades:

   - Creación de matrices y casos de prueba para pruebas funcionales.
   - Creación de matrices y casos de prueba de negocio.
   - Registrar y validar la corrección de errores reportados en las herramientas de seguimiento de defectos.
   - Reportar avance de pruebas al líder de pruebas asignado al proyecto.
   - Mantener una comunicación constante con el líder de pruebas asignado.
   - Emitir Vo.Bo. cuando la aplicación ya se encuentre estable en el ambiente de pruebas de aceptación.
   - Seleccionar y ejecutar casos de prueba representativos de negocio una vez que la aplicación se encuentre en ambiente de producción para asegurar que la integridad del sistema se mantiene.

5.3. NUEVO PROCESO DE PROCESO DE PRUEBAS

En esta sección se describirán las fases del nuevo proceso de pruebas, considerando el siguiente diagrama del modelo diseñado:

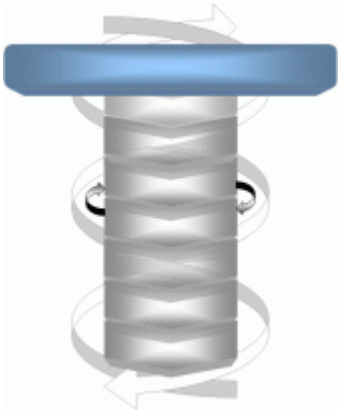


5.3.1. Administración de pruebas

Objetivo general

Organizar, manipular y controlar los artefactos generados dentro de un proyecto tales como requerimientos, planes, documentación, scripts y resultados de pruebas (por mencionar algunos activos del testware). Esta actividad se realiza para contar con accesibilidad y reusabilidad.

Introducción



"La administración de pruebas es un área de conocimiento que cuenta con habilidades, herramientas y técnicas utilizadas para alcanzar los objetivos del proyecto, y aplica los principios de ingeniería de software, implementación y seguimiento".

La idea es utilizar principios de ingeniería de software para poder probar aplicaciones en menos tiempo y así obtener la calidad esperada. Esta actividad tiene como base conceptos de colaborar y compartir información. Actividades como planeación, diseño y ejecución de pruebas representan una cantidad considerable de trabajo, pero este esfuerzo será recompensado cuando todos los activos de pruebas son compartidos por los ingenieros de pruebas, desarrolladores y administradores del mismo modo.

Proceso, ¿Qué debo de hacer?

Es importante mencionar y saber que la administración de pruebas no es realizada sólo por un gerente de pruebas o un líder de pruebas, es decir, que no se refiere a las actividades especificas de un rol sino a un área que combina habilidades y experiencia, por lo que:

"La administración de pruebas, como actividad, es realizada por todos los miembros del equipo y no sólo por el gerente de pruebas o el líder de pruebas".

A continuación se describen algunas actividades que se deben de llevar a cabo en esta fase:

- Control de pruebas / seguimiento / control de cambios. Integrar y sincronizar planes de trabajo, dar seguimiento a actividades además de controlar los cambios que se vayan generando en el proyecto.
- Control del alcance. Definir y plantear el alcance de las pruebas.
- Control de actividades. Se deben estimar y asignar actividades de acuerdo a los miembros del equipo así como también se debe generar la secuencia en la cual dichas actividades serán llevadas a cabo.
- Administración de recursos. Es necesario mantener un control y administración de todos los recursos involucrados en la ejecución de las pruebas, ya sean humanos o materiales. Algunos ejemplos de esta administración son la solución de conflictos, selección del equipo.
- Comunicación. Mantener siempre un plan de comunicación con el equipo, con clientes internos, usuarios, desarrolladores, administradores, gerentes, etc., así como siempre reportar el estatus de las pruebas.
- Administración de riesgos. Se deben documentar y escalar los riesgos que comprometan las fechas de entrega, la calidad del sistema, la calidad de pruebas, o cualquiera de los objetivos planteados.

Aunque la administración de riesgos es responsabilidad del líder de pruebas, también es responsabilidad del staff de pruebas, el reportar todo aquel riesgo que sea identificado.

Riesgos

Los riesgos surgen a partir de la incertidumbre que exista en las decisiones o resultados de un proyecto. Usualmente el concepto de riesgo se asocia con la pérdida potencial en costos, control, funcionalidad, calidad o tiempos, pero antes de comenzar con la administración de riesgos es necesario conocer qué es un riesgo:

"Un riesgo se define como cualquier evento o condición que pueda tener un impacto negativo o positivo en los resultados de un proyecto".

Se debe saber que un riesgo es diferente a un problema o a un issue ya que los riesgos se refieren a pérdidas o diferencias potenciales en los resultados a futuro, mientras que los problemas o issues son condiciones o estados que existen dentro del proyecto en el presente. Los riesgos se pueden llegar a convertir en problemas si no se administran y enfocan correctamente.

A la administración de pruebas le corresponde identificar, analizar y escalar los riesgos identificados de manera proactiva, de esta forma se llega al objetivo de maximizar los impactos positivos (oportunidades) y minimizar los impactos negativos (pérdidas) asociados con el proyecto en curso.

Para poder identificar, analizar y escalar los riesgos de una forma correcta se deben tomar en cuenta los siguientes aspectos:

- Anticipar los problemas en lugar de sólo lidiar con ellos cuando se presenten.
- Identificar las causas en vez de sólo ver los síntomas.
- Contar con planes de contingencia antes de que ocurra un problema.
- Usar medidas preventivas tanto como sea posible.

Para documentar lo anterior es necesario que al identificar un riesgo, éste contenga la siguiente información:

- *ID*. Identificador único con el que se puedan localizar fácilmente los riesgos.
- *Fecha*. Fecha en la cual se encontró o identificó el riesgo.
- *Riesgo*. Una descripción detallada de éste.
- *Consecuencia del riesgo*. Descripción del problema en el que se podría convertir el riesgo en caso de no mitigarlo.
- *Acciones propuestas*. Dar, siempre que sea posible, sugerencias sobre como mitigar el riesgo.
- *Contingencia*. Acciones que se llevarían a cabo en caso de que no se pueda lidiar con el riesgo o que éste ya se haya convertido en un problema.
- *Responsable*. Nombre de la persona más indicada para resolver o tomar responsabilidad por el riesgo reportado.
- *Fecha límite de resolución*. Fecha máxima en la cual se tiene que dar una solución al riesgo.

Métricas

Las métricas son un buen medio para entender, monitorear, controlar, predecir y por lo tanto probar el desarrollo de una aplicación, dicha medición tiene como objetivos:

- Entender qué ocurre durante el desarrollo del sistema.
- Controlar qué es lo que ocurre en los proyectos.
- Incrementar la calidad de las aplicaciones mediante la mejora continua de los procesos.

Los tipos de métricas que se pueden obtener a partir del testware generado para las distintas aplicaciones probadas son las siguientes:

Con los casos de prueba:

- Lo que se ha probado.
- En cuánto tiempo se ha probado.
- Lo que falta por probar.
- Los casos exitosos.
- Los casos no exitosos.

Y con respecto a los errores:

- Los errores corregidos.
- Los que faltan por corregir.
- Errores revisados.
- Errores no detectados en el proceso de pruebas.
- Tiempo en que se tarda un error en ser corregido.
- Tipo de errores encontrados.
- Errores por desarrollo.
- Tipo de funcionalidad con errores críticos.

Con los datos y mediciones anteriores se puede recabar información muy valiosa, ya que brinda las bases necesarias para planear y mejorar la aplicación del proceso de pruebas en futuros proyectos.

Reporte de pruebas realizadas

Durante las diferentes etapas del proceso se van generando entregables, los cuales sirven tanto para realizar las pruebas como para generar los diferentes reportes que permiten identificar el avance que se lleva.

Estos reportes son parte importante en la comunicación con los gerentes y desarrolladores ya que se puede conocer “lo probado”, “lo no probado”, “lo que falta por entregar”, entre otros.

La generación de reportes sirve principalmente para:

- *Conocer el estatus de la aplicación*. Saber exactamente qué partes del sistema ya fueron verificadas por el proceso de pruebas y qué porcentaje de éste cumple con los requerimientos funcionales. Además, se puede identificar qué parte de la aplicación o módulos hace falta entregar por el equipo de desarrollo.
- *Conocer el estatus de las pruebas*. Saber qué parte del testware diseñado ha sido aplicado y cuánto falta por verificar.
- *Conocer los errores*. Los reportes de errores permiten tener una visión más clara de la gravedad o severidad de éstos, quién los ha generado y cuánto tiempo tomó corregirlos.
- El contar con reportes que contengan información a nivel ejecutivo proporciona mejores bases a los gerentes para la toma de decisiones y sobre qué acciones implementar de acuerdo a la calidad actual de la aplicación.

Control de versiones

Un control de versiones es un registro donde se tienen documentadas las características de cada versión, release o build liberado por el área de desarrollo. A cada versión se le ejecutan los diferentes artefactos de prueba y se van encontrando diferentes errores.

El registro de las características nos permite conocer toda aquella funcionalidad no liberada, funcionalidad agregada, errores corregidos; de tal forma que con cada versión se hace más robusta la aplicación y se cuenta con el “historial” de su avance.

El control de versiones debe contener ciertos elementos básicos:

Elemento	Descripción
Número de versión / build / release	Número con el que se identifica la versión que está en proceso de pruebas
Fecha de liberación por el área de desarrollo	Fecha en la cual el área de desarrollo terminó de hacer las pruebas unitarias y se encuentra en condiciones de pasar la aplicación al equipo de pruebas
Fecha de liberación por el equipo de pruebas	Fecha en la cual el equipo de pruebas terminó de ejecutar su testware y está listo para pasar a las pruebas de aceptación.
Funcionalidad agregada	Lista de la funcionalidad agregada a la versión
Funcionalidad no liberada	Lista de funcionalidad faltante
No. de errores encontrados	Número de errores nuevos
No. de errores por corregir	Número de errores que todavía existen en la versión
No. de errores cancelados	Número de errores que por algún motivo fueron cancelados
No. de errores cerrados	Número de errores corregidos y que han pasado la verificación de calidad
Estatus	Dictamen de la versión, ¿debe ser liberada o no?
Comentarios	Campo para agregar algún comentario u observación sobre la versión registrada

Bitácora de pruebas

Durante las distintas fases de prueba en las cuales las aplicaciones son revisadas, todos los miembros del equipo llevan a cabo diferentes actividades, ya sean planeadas o no. Dichas actividades deben de ser registradas.

El documento destinado a registrar las actividades debe de contener:

- Fecha
- Actividad realizada
- Módulo / submódulo / proyecto
- Observaciones

La información servirá para conocer en qué está trabajando cada miembro del staff de pruebas, problemas que se han presentado, y conocer el tiempo que se invierte en probar cada módulo.

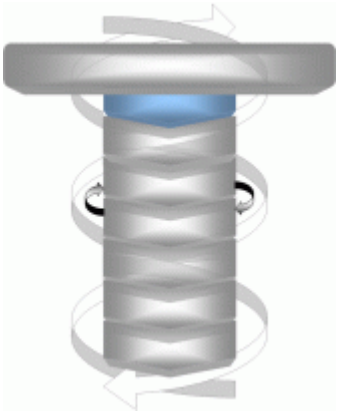
Entregables (registros generados)

- Control de riesgos
- Control de versiones
- Métricas
- Bitácora de pruebas
- Reportes

5.3.2. Análisis y planeación de pruebas

Objetivo general
Análisis del proyecto, determinación del alcance, planeación de las pruebas y configuración del ambiente operativo de la aplicación.

Introducción



Las pruebas de software tienen como objetivo lograr un producto de calidad que cumpla con las especificaciones del software, y que sea un producto con el cual el cliente quede satisfecho. Con base a lo anterior es indispensable estudiar y analizar la información existente del proyecto, para determinar el alcance y esfuerzo de las pruebas, así como planearlas de acuerdo a la complejidad y necesidades del proyecto; de aquí surge la etapa de "análisis y planeación de pruebas".

Este análisis y planeación se basa en una pregunta fundamental:

¿Qué voy a probar?

La respuesta se encuentra en 2 elementos principales:

- Requerimientos funcionales: El requerimiento de un sistema o sistemas que ejecuten una función en específico.
- Requerimientos no funcionales: carga, estrés, usabilidad, volumen, documentación, desempeño, seguridad.

Otro punto importante dentro de esta fase, es el controlar la configuración del ambiente de pruebas. Durante mucho tiempo hemos visto que los profesionales de pruebas utilizan la máquina que les fue asignada para probar código fuente (.exe's), agregar y eliminar dll's, manipular bases de datos, instalar y desinstalar setup's, realizar conexiones con el servidor, etc. Pero resulta que donde se realizan estas pruebas se tiene instalado cuanto software nos podamos imaginar, dando como resultado que el sistema funcione o no debido a las diversas aplicaciones instaladas o desinstaladas. En esta guía mostraremos algunas recomendaciones para tener un correcto control de esta configuración.

Finalmente, es necesario planear a detalle el tiempo que nos llevará probar la aplicación, así como detallar cada una de las actividades por lo que también en esta fase se hablará del plan de trabajo desde el punto de vista de las pruebas de software.

Proceso, ¿Qué debo hacer?
En esta fase se revisará lo que se tiene que probar y con base en eso, se realizará la planificación de las pruebas, de acuerdo a la prioridad y complejidad de los casos de uso o escenarios de prueba que deben contemplarse dentro de la aplicación de las pruebas.

En ocasiones, de acuerdo a los planes de trabajo ya definidos por los proyectos, es necesario acoplar el tiempo de pruebas a éste. Es recomendable que siempre, durante la definición del plan general del proyecto, esté presente un representante de pruebas para poder dar tiempos aproximados y reales de lo que tomaría cada una de las fases, de acuerdo a la complejidad del sistema.

Desde el análisis y la planeación pueden identificarse errores en el contenido de la documentación del proyecto, tales como:

- Comprensión errónea de los requerimientos.
- Requerimientos no contemplados.
- Errores en la usabilidad, revisando el diseño del sistema.

En esta fase esta divida por diferentes actividades:

1. Analizar la documentación existente
2. Determinar el alcance de las pruebas
3. Realizar pruebas de requerimientos
4. Definir escenarios de prueba
5. Generar plan de trabajo
6. Configurar el ambiente de pruebas

1. Análisis de la documentación existente

Objetivo: Analizar la funcionalidad del sistema: explicación de los requerimientos funcionales por parte de los analistas del negocio y/o expertos, así como el estudio de la documentación existente.

Existen dos escenarios dentro de los proyectos que afectan directamente a esta actividad, uno donde existe documentación del sistema y otro en el que no existe nada documentado. Para cada uno de los casos, los pasos a seguir se describen a continuación.

Escenario 1: Existe documentación del proyecto.
Es importante solicitar al líder o responsable del proyecto la documentación existente. Esta documentación puede ser variada, lo ideal es contar con los siguientes artefactos:

- Documentación de los requerimientos de negocio
- Especificación de diseño
- Especificación funcional
- Flujo de eventos
- Priorización de requerimientos
- Plan de trabajo
- Documento de riesgos
- Prototipo
- Manual del administrador
- Manual de usuario
- Glosario de términos

Se deberá leer toda la documentación existente, comprender completamente de qué se trata la aplicación, como funciona y que es lo que quiere el cliente.

Al terminar este análisis y haber comprendido toda la documentación proporcionada, es importante tener pláticas con los expertos (líder de proyecto, analistas, desarrolladores y usuarios) para afianzar el conocimiento adquirido.

Escenario 2: No existe documentación del proyecto.
En caso de que no exista documentación del proyecto que nos pueda explicar, la forma en la cual funciona el sistema y cuáles son los requerimientos del cliente (¿Qué es lo que debe hacer? y/o ¿Qué es lo que voy a probar?), es necesario llevar a cabo reuniones con los expertos del sistema.

Es recomendable además de esta explicación, y en caso de que exista ya la aplicación desarrollada o un prototipo de esta, una capacitación sobre la misma para comprenderla a fondo y poder probarla de manera correcta y eficiente.

Deberá documentarse todo el conocimiento adquirido de forma entendible, además, deberá ser validado con el cliente, el líder de proyecto o alguno de los expertos para verificar que sea correcto lo que se entendió del sistema.

Recomendación: La manera de como se recomienda documentar este conocimiento es a nivel de procedimientos de pruebas en su modalidad de procedimiento funcional.

2. Determinación del alcance de las pruebas

Objetivo: Analizar la funcionalidad del sistema: Explicación de los requerimientos funcionales por parte de los analistas de negocio y/o expertos, así como el estudio de la documentación existente.

En esta actividad se deben de contemplar los siguientes puntos:

1. Identificar la información existente del proyecto y los componentes de software que deberán ser probados.

2. Identificar y enlistar los requerimientos de alto nivel que serán probados.

3. Recomendar y describir las estrategias de pruebas a ser utilizadas.
    - Proceso a seguir: Proceso de pruebas estándar de la organización.
    - Establecer las etapas, técnicas y tipos de pruebas a aplicar.
    - Identificar los recursos necesarios y proporcionar una estimación del esfuerzo requerido para realizar las pruebas, ya sean recursos materiales, como podría ser un laboratorio de pruebas, identificando las características de éste, así como recursos humanos para poder llevar acabo el proyecto. Todo esto en resumen es: ¿Qué requiero para poder probar sin problemas? ¿Necesito más gente? ¿Otra máquina?, etc.

4. Establecer las herramientas a utilizar. Este punto aplica en el momento que se desea y se cuenta con los elementos necesarios para automatizar ciertos procesos de pruebas. Es importante realizar un análisis de qué herramienta de automatización es la más adecuada para el proyecto, así como listar las necesidades o requerimientos especiales de la herramienta.

5. Definir requerimientos especiales, esto es, si para poder probar la aplicación, por ejemplo que requiera generar reportes en una impresora determinada, sería un requerimiento especial, ya que para poder probar esta función se necesita contar con la impresora y así verificar que lo haga correctamente.

El responsable de las pruebas elaborará un artefacto conocido como "especificación de pruebas" en donde determina el propósito, alcance, los tipos de pruebas a realizar, herramientas a utilizar, requerimientos a probar, estrategia de pruebas, entre otros.

Los elementos que contiene este artefacto se describen a continuación:

Elemento	Descripción corta
Introducción 1. Propósito 2. Antecedentes 3. Alcance 4. Identificación del proyecto	Resumen tanto del proyecto como de la organización ¿Con qué información cuento?
Requerimientos a probar 1. Casos de uso, requerimientos funcionales 2. Requerimientos no funcionales	¿Qué voy a probar?
Estrategia de pruebas 1. Proceso de pruebas 2. Etapas, técnicas y tipos de pruebas. 3. Herramientas	¿Cómo lo voy a hacer?
Recursos 1. Recursos de sistema 2. Recursos humanos	¿Qué necesito?
Actividades del proyecto	¿Cuánto tiempo necesito y que es lo que haré?
Criterios de aceptación del sistema	¿Qué criterios deben cumplirse?, Estos serán validados por el cliente para la liberación del sistema

3. Realizar pruebas de requerimientos

Objetivo: Evaluar la consistencia, completitud y factibilidad de los requerimientos, tanto individualmente como en conjunto.

Un requerimiento es un rango de instrucciones abstractas de alto nivel de un servicio o de un sistema, limitado a detallar una especificación funcional. Las pruebas de requerimientos se realizan durante el análisis de la documentación del proyecto, esto con el fin de identificar las deficiencias dentro de la misma, requerimientos incorrectos o faltantes.

Lo que se debe realizar, es leer los casos de uso de la aplicación, donde cada uno de ellos describe una tarea que debe llevar a cabo un actor (ya sea el sistema, usuario); por lo que pueden generarse casos de prueba para validar que las tareas se realicen según lo indica el caso de uso. El caso de uso es una narrativa en lenguaje informal, con los inconvenientes que esto trae. Adicionalmente el caso de uso describe un flujo de control que describe qué pasos pueden repetirse, cuáles son opcionales, qué cursos alternos hay, flujos que deben probarse.

Se proponen que se generen los siguientes casos de prueba:

- Casos correspondientes al curso normal del caso.
- Casos correspondientes a cursos excepcionales.
- Casos que surgen de requerimientos específicos a un objeto del caso de uso.
- Casos asociados a la prueba de características descritas en documentos asociados al caso de uso.

4. Definir escenarios de prueba

En la definición de escenarios de prueba (sobre todo para pruebas diferentes a las funcionales), se requiere de la participación del cliente para poder definirlos. Estos escenarios son la base para la ejecución de las pruebas, por lo mismo se requiere que el experto defina de forma general qué es lo que se debe probar, además de incluir algunas características como son:

- Frecuencia
- Velocidad
- Tiempo de respuesta
- Capacidad
- Número de accesos concurrentes
- Cantidad de usuarios

Es importante, como recomendación al cliente, que se realice teniendo el punto de vista de los expertos en la aplicación, esto es líder de proyecto, desarrolladores y usuarios.

Es necesario recalcar al cliente que esta definición es prioritaria debido a que en base a esta se realizará la planeación de las pruebas.

Todo el diseño de las pruebas se basará en dichos escenarios aunque están sujetos a cambio, debido a que en ocasiones la aplicación puede no soportar la carga con la cual se planearon originalmente las pruebas.

Se requiere priorizar a cada uno de los escenarios para poder asignar los tiempos a cada uno de ellos, tanto para el diseño como para la ejecución de las pruebas, como se muestra en la siguiente tabla:

ESCENARIO	**ATRIBUTOS**			
	Prioridad	**Riesgo**	**Estabilidad**	**Complejidad**
		(Alto, Medio, Bajo)	*(Alta, Media, Baja)*	*(Alta, Media, Baja)*

5. Generar plan de trabajo

Objetivo general: Delimitar las tareas y tiempos necesarios para el esfuerzo de pruebas.

Adicional al plan general de proyecto (especificación de pruebas), se requiere generar un plan de las pruebas a realizar para poder llevar un mejor y mayor control sobre las mismas. Es necesario asignar:

- Actividades: Detallar las actividades a realizar durante el proceso de las pruebas del proyecto.
- Tiempos: Asignar los tiempos de cada una de las actividades, definiendo fecha inicio y fin de cada una de ellas.
- Recursos: Definir los recursos que estarán involucrados en las actividades y los cuales serán responsables de ejecutar cada una de ellas.

Es necesario que el plan de las pruebas se lleve a cabo como está definido, esto es, respetar los tiempos y fechas establecidos. En caso de que se identifique algún riesgo que interfiera con el desarrollo del plan de trabajo, es necesario comentarlo y buscar alternativas para mitigarlo, modificar fechas de entrega, asignar más recursos, etc.

El ingeniero de pruebas necesita analizar el plan de trabajo del proyecto para diseñar su propio plan de pruebas. Le servirá el plan del proyecto para saber cuánto tiempo le puede dedicar a cada una de las actividades de pruebas, el orden y fechas en que se tiene planeado terminar cada una de las funcionalidades o casos de uso del sistema de forma tal que él planeé en el mismo orden el diseño de pruebas y la aplicación de las mismas.

El plan de trabajo general le servirá también para conocer el número de casos de uso y los programadores que se encargarán de desarrollarlos, esto con el fin de llevar un control de los errores que se irán encontrando a lo largo de la fase de pruebas y conocer a los responsables para su corrección.

Este formato nos ayudará a llevar un control en cuanto a los tiempos que se tiene para realizar las pruebas. Este documento se basará en el tiempo que se tenga designado para el proceso de pruebas en el plan de trabajo general del proyecto.

Para la estimación de pruebas se recomienda seguir los siguientes procesos de estimación:

- ***Proceso de estimación tradicional:***
  - Tiempo de pruebas: Tiempo de desarrollo / 3
  - Ingenieros de pruebas: Desarrolladores / 2
  - Costo de pruebas: Tiempo de pruebas x Ing. de pruebas x Costo

- ***Proceso de estimación propuesto:***
  - No. de casos de prueba: Número de requerimientos x 4.53
  - Fase de diseño: Número de casos de prueba / 14.47
  - Fase de ejecución: Número de casos de prueba / 10.20
  - Pruebas de regresión: Tiempo de la fase de ejecución x 0.35

Los siguientes valores fueron tomados del método de “estimación estándar”:

- 4.53 = Cantidad promedio de casos de prueba por cada requerimiento.
- 14.47 = Número de casos de prueba promedio diseñados por día.
- 10.20 = Número de casos de prueba promedio ejecutados por día.
- La fase de regresión se calcula obteniendo el 35% del total de ejecución de pruebas.

Estos tiempos son por ingeniero de pruebas. Los valores se deben ir ajustando de acuerdo a las necesidades de cada organización, en base a las estimaciones de proyectos ejecutados.

6. Configuración del ambiente de pruebas

Objetivo general: Superar las expectativas de nuestros clientes, garantizando que el producto que les entregamos ya fue probado en un ambiente real, muy similar o idéntico al ambiente de producción de los mismos.

Actualmente la ejecución de pruebas se puede hacer de muchas formas y hacerlo muy bien, pero ¿por qué cuando se instala con el cliente el sistema falla?, ¿por qué el rendimiento no es el esperado?, esto es debido a que el ambiente en el que probamos a veces no se parece ni por mucho al ambiente real del cliente, lo cual es un riesgo que no nos podemos dar el lujo de correr.

El propósito de este artefacto es controlar y realizar las pruebas en diferentes ambientes, teniendo así un mejor manejo del hardware de su equipo, así como una mejor organización y control de versiones.

También ayudará a los líderes de proyecto a llevar un mejor control de los setup's generados y los mostrados al cliente, además de mejorara el panorama del desarrollo real del sistema.

En este artefacto se capturarán las características de hardware y software que tiene la computadora en la que se realizarán las pruebas y de los requerimientos mínimos que necesita el sistema para poder ejecutarse, así se garantiza que las pruebas que se realicen y los resultados que se obtengan serán los mismos que cuando el sistema sea operado por el cliente.

La configuración del equipo de pruebas se refiere a características como las siguientes, que podrían o no aplicar, dependiendo de los requerimientos especificados:

- Tipo de procesador de la PC
- Velocidad del procesador
- Memoria mínima en RAM
- Espacio en disco duro
- Tipo y resolución del monitor
- Sistema operativo y su versión
- Librerías especiales utilizadas y su versión (ADO, ODBC, OCXs, controles de VB, etc.)
- Base de datos y su versión

- Volumen de datos requeridos en la base de datos
- Tipo de red
- Y cualquier otro equipo y características adicionales especificadas por el cliente para que la aplicación desarrollada funcione de acuerdo a lo establecido.

Es necesario preparar e instalar todo aquello que simulará el ambiente de operación para una aplicación específica. Por lo tanto la configuración del ambiente de pruebas se establece para cada proyecto.

Es importante hacer notar que no es indispensable realizar las pruebas exactamente sobre un equipo con las características especificadas, sino que puede ser sobre equipo no tan eficiente como el especificado, y así, si los resultados de las pruebas son satisfactorios, entonces podemos garantizar que lo serán más sobre un equipo mejor (como el especificado).

Las ventajas que se identifican a primera instancia son las siguientes:

- Realización de pruebas en un ambiente real.
- Disminución de los errores encontrados en el ambiente del cliente.
- Mayor satisfacción y seguridad por parte de nuestros clientes.
- Ahorra tiempo a los desarrolladores sobre todo en la fase de transición.
- Se identifican errores o posibles errores anticipadamente.

Para hacer la configuración del ambiente de pruebas se puede utilizar cualquiera de los siguientes métodos:

1. Uso de máquinas virtuales
   El crear máquinas virtuales independientes en un escritorio Windows, nos permite simular una PC completa y real en una misma máquina, pudiendo accederlas de manera fácil y rápida para poder aplicar las pruebas y representar más ambientes de prueba. Algunas de las herramientas para poder generar máquinas virtuales son las siguientes:

   - MS Virtual PC
   - VMware
   - Connectix

2. Particionar el disco duro y contar con imágenes de diferentes sistemas operativos
   Dentro del mismo equipo de pruebas se pueden crear particiones, a las cuales una vez que está instalado el software base como es el sistema operativo, service pack's, paquetería básica como Office, etc., se usan aplicaciones como Ghost para sacar una imagen del disco en este estado, así, cada vez que se requiera instalar la aplicación en un ambiente "limpio", sólo se aplica la imagen al disco duro y queda listo para continuar probando.

   Dicha configuración consiste básicamente en que el ingeniero de pruebas tenga en su equipo de trabajo tres particiones en el disco duro, de tal forma que una esté asignada al desarrollo, otra a las pruebas y otra a la información.

Cabe mencionar que para utilizar este método se requiere que la PC tenga el espacio y la memoria suficiente para soportar este esquema además de conocer la antigüedad de cada sistema operativo, ya que se tienen que ir instalando en ese orden, del más “viejo” al más “nuevo”.

Entregables (registros generados)

- Especificación de pruebas
- Plan de trabajo
- Configuración del ambiente de pruebas

5.3.3. Diseño de pruebas

Objetivo general

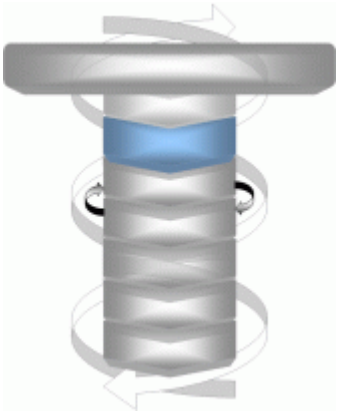
Identificar y comunicar la información necesaria sobre como implementar los casos de prueba de forma apropiada y consistente, especificando los refinamientos necesarios sobre el enfoque general reflejado en el plan e identificar las características que se deben probar con este diseño de pruebas.

Propósito:

- Identificar un conjunto de casos de prueba que puedan ser verificados y construidos.
- Identificar procedimientos de prueba que muestren como se realizarán los casos de prueba.

Nota: El equipo de pruebas decidirá las herramientas a utilizar de acuerdo a las necesidades del proyecto.

Introducción



Dentro del proceso de pruebas existe una etapa en donde toda la información analizada y estudiada se convierte en artefactos de prueba y se le da respuesta a preguntas como las siguientes: ¿Cómo lo voy a probar?, ¿Qué voy a usar para asegurarme que la aplicación funcione correctamente?, ¿Qué es lo más adecuado para probar este sistema?, etc.

El objetivo de todo diseño de pruebas es plasmar todos aquellos elementos sobre como debería y como no debería de funcionar una aplicación usando diferentes componentes del testware, siempre basados en la especificación funcional definida previamente.

Proceso, ¿Qué debo de hacer?

Existen diversas formas de crear y aplicar artefactos de pruebas dependiendo de la funcionalidad del sistema que se quiera probar.

A continuación se da una explicación general de los elementos del diseño de pruebas:

- **Procedimientos de pruebas:** Un procedimiento de pruebas tiene como objetivo especificar los pasos para la ejecución de un conjunto de casos de prueba o, más generalmente, los pasos utilizados para analizar un elemento software con el propósito de evaluar un conjunto de características del mismo.

  Estos artefactos están divididos principalmente por los tipos de pruebas a aplicar:

  - Pruebas funcionales.
  - Pruebas de performance, usabilidad, instalación, etc.

  Los procedimientos para pruebas de funcionalidad se refieren a una explicación a todo detalle de cada uno de los casos de uso (requerimientos funcionales) que contiene el sistema.

  Para estas pruebas se utilizará un procedimiento por cada requerimiento funcional complejo (entendiéndose por complejo un caso de uso muy grande o que para su entendimientos se requiera de mayor detalle).

  Aquí nos enfocaremos a una explicación de manera particular de lo que debe y como debe de realizar cada uno de los procesos la aplicación. Es como un manual de usuario pero a más alto nivel. Este documento llevará una tabla de validaciones donde se mencionará las características de cada uno de los campos (dato requerido, dato opcional, dato de consulta, dato editable). También se podrán insertar accesos a base de datos, tablas, etc.

  Se recomienda también anexar una representación gráfica de la manera como se comportará el caso a nivel pantallas y/o procesos. Por ejemplo:

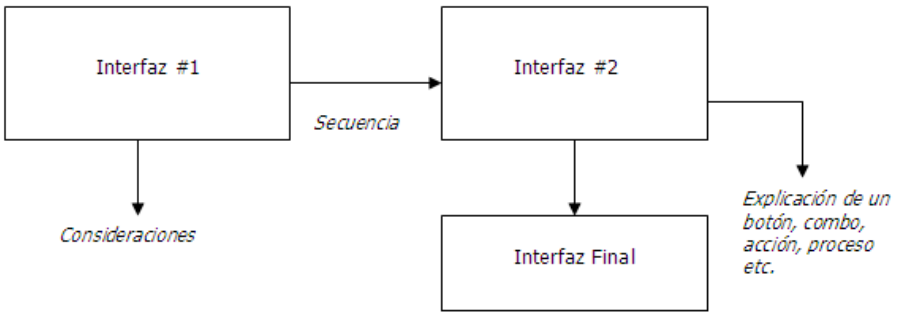


Tabla de Validaciones

Dato Requerido
Dato Opcional
Dato Editable
Dato Consulta

			Nombre del Campo	Tipo	Longitud

En el caso de los procedimientos para otros tipos de pruebas (diferentes a las funcionales) los detallaremos de la siguiente manera.

1. Definición del tipo de prueba a aplicar.
2. Objetivo de la prueba.
3. Técnica a utilizar.
4. Criterio de aceptación.
5. Consideraciones especiales.

- **Casos de pruebas:** Conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular.

  Propósito: Identificar y comunicar las condiciones que serán puestas en ejecución y que son necesarias para verificar y aceptar un requerimiento funcional. También es posible diseñar de esta forma requerimientos no funcionales.

  Para poder comenzar a diseñar casos de prueba se debe de considerar lo siguiente:

  - Todos los casos de uso tienen casos de prueba.
  - Para cada caso de uso, siempre existen por lo menos un caso de prueba exitoso y uno no exitoso.
  - Dentro del proceso de pruebas se contemplan dos tipos de casos de pruebas, los específicos (los cuales se refieren a un proceso particular de una funcionalidad) y los generales (que son casos de prueba que aplican para más de un proceso, caso de uso, módulo, etc.)

  Dentro del diseño de casos de prueba podemos identificar principalmente las siguientes actividades:

Responsable	Actividad
Usuario, BA, STE	**Identificar los casos de prueba exitosos y no exitosos.** • Un *caso de prueba* exitoso es aquel que al concluir cumple con el objetivo del caso de uso (test to pass). • Por el contrario, un *caso de prueba* no exitoso, es aquel que al concluir *no* cumple con el objetivo del caso de uso (test to fail).
Usuario, BA, STE	**Describir brevemente el objetivo del caso de prueba.** Es una breve explicación (descripción corta) de lo que se busca al efectuar ese caso de prueba. Esta descripción también se conoce como pregunta corta. Por ejemplo: ¿Qué pasa si...?
Usuario, BA, STE	**Delimitar los pasos a seguir.** Estos pasos son indispensable que para poder ejecutar lo que marca la "condición a ser probada".
Usuario, BA, STE	**Identificar los resultados esperados y obtenidos del caso de prueba.** Es el estatus en el que se encuentra el sistema después de ejecutar el caso de prueba y el estatus que esperábamos.

En los casos de prueba se utilizan los siguientes elementos:

Titulo de la columna	Definición
No.	Número secuencial del caso de prueba en este documento
Condición a ser probada	Funcionalidad específica a ser probada
Detalles de ejecución	Pasos a seguir para completar el caso de prueba
Resultados esperados	¿Qué es lo que debe ocurrir?
Resultados reales	¿Qué es lo que ocurrió en realidad?
¿Prueba exitosa?	¿La aplicación funcionó como se esperaba? (S: Sí; F: Falló, SP: Sin Probar. Si es F, se debe describir la falla en la herramienta de errores, o en una plantilla)

- **Matrices de pruebas:** Combinatoria de todas las variantes de los casos de prueba con cada uno de sus valores, de tal forma de que no haya algún escenario sin validar por más remoto o poco probable que suceda.

  Las matrices se elaboran para garantizar pruebas robustas para todos los casos de prueba exitosos o no exitosos, y que la modificación de alguna especificación no represente demasiada inversión de tiempo en la modificación de las matrices.

  Estas matrices vienen complementando los casos de prueba y tendrán que contener todas y cada una de las opciones en las que se pueden ver involucrados nuestros casos de prueba, ya que a medida que sean más completos nuestros casos de prueba, y más robustas nuestras matrices, la posibilidad de que el usuario encuentre un error tiende a cero.

  A continuación se enlistan los pasos para completar este artefacto satisfactoriamente.

  1. Identificar las variantes que afectan a los casos de prueba exitosos y no exitosos. Este es un paso crítico en el procedimiento, y lo más importante es que el ingeniero de pruebas esté consciente que las variantes que identifique en un principio no forzosamente son las ideales para generar toda la combinatoria de matrices.

     Es muy importante que el ingeniero de pruebas empiece a generar la combinatoria (ver punto 3) para darse cuenta si las variantes iniciales fueron las idóneas. Este es el punto más elaborado, el cuál exige más del ingeniero de pruebas. Para identificar una variante hay que identificar los puntos (procesos, campos de texto, option buttons, combos, perfiles de usuario, conexiones, hardware, software, interfaz, etc.) que puedan tomar 2 o más valores.

  2. Generación de combinaciones basándose en las variantes y sus valores. Tomando en cuenta todas las variantes hay que generar una combinatoria, identificando cual corresponde a un caso de prueba exitoso y cuál a un no exitoso. Hay que contemplar los valores posibles para cada variante así como sus límites (reglas del negocio).

NOTA: Es muy importante identificar cuando cierta combinatoria anula muchas más, de esta forma evitamos el retrabajo y podemos garantizar que el resultado de la prueba será el mismo y que no se está omitiendo ningún escenario.

3. Asignar a los diferentes valores que puede tomar cada combinatoria, un dato distinto y bajo circunstancias distintas pero que cumplan con las limitantes planteadas anteriormente. De esta forma no solo se prueba una vez cada caso de prueba sino que se ejecuta el mismo caso de prueba bajo distintas condiciones.

4. Llevar el registro de cuáles son las matrices que ya fueron validadas con éxito y cuáles no pasaron con éxito la prueba para registrarlas en el control de errores.

   Finalmente a manera de resumen, la manera como el nuevo proceso de pruebas propone que se diseñen estas matrices es en tres pasos muy sencillos:

   - Determinar las variantes y valores para un caso de uso.
   - Delimitar las combinaciones que pueden tener entre ellos.
   - Documentar los datos utilizados para las pruebas.

Por ejemplo, pensemos en el caso de un retiro de efectivo de un cajero automático.

Paso 1

Variantes	Valores
Tipo de tarjeta	Doméstica Red
Tipo de monto a retirar	Fijo Variable
Límites	>= 50 <= 3000
Saldo en cuenta	Saldo > Monto Saldo < Monto Saldo = Monto
Efectivo en cajero	Efectivo >= Monto Efectivo < Monto
Monto retirado en el día	0 < 3000 = 3000,
Denominación de billetes	50 100 200 500
Número de billetes más pequeños	Suficiente e Insuficiente

Paso 2

Exitoso / No exitoso	Tipo de tarjeta	Tipo de monto	Límites	Monto R D + monto solicitado	Saldo cuenta	Efectivo en cajero	Denominación	Billetes más pequeño
Exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$50	Suficiente
No exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$50	Insuficiente
Exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$100	Suficiente
No exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$100	Insuficiente
Exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$200	Suficiente
No exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$200	Insuficiente
Exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$500	Suficiente
No exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	>= monto	$500	Insuficiente
No exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto	< monto	N/A	N/A
No exitoso	Doméstica	Fijo	>=50 y <=3000	$3,000	N/A			
No exitoso	Doméstica	Fijo	>=50 y <=3000	< $3,000	Saldo < Monto			
Exitoso	Doméstica	Variable	>=50 y <=3000	< $3,000	Saldo >= Monto			
No exitoso	Doméstica	Variable	>=50 y <=3000	$3,000	N/A			
Exitoso	Doméstica	Variable	>=50 y <=3000	< $3,000	Saldo < Monto			
No exitoso	Doméstica	Variable	< 50 o >3000	N/A	N/A			
Exitoso	Red	Fijo	>=50 y <=3000	< $3,000	Saldo >= Monto			
No exitoso	Red	Fijo	>=50 y <=3000	$3,000	N/A			
No exitoso	Red	Fijo	>=50 y <=3000	< $3,000	Saldo < Monto			
Exitoso	Red	Variable	>=50 y <=3000	< $3,000	Saldo >= Monto			
No exitoso	Red	Variable	>=50 y <=3000	$3,000	N/A			
No exitoso	Red	Variable	>=50 y <=3000	< $3,000	Saldo < Monto			
No exitoso	Red	Variable	< 50 o >3000	N/A	N/A			

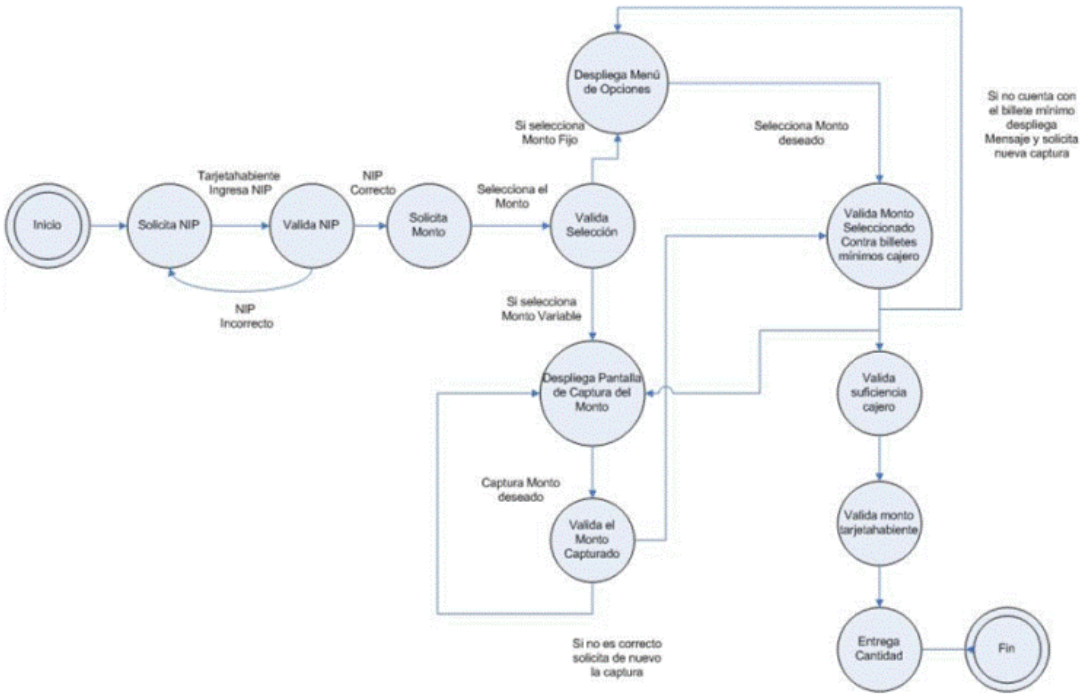
Paso 3

Exitoso / no exitoso	Tipo de tarjeta	Tipo de monto	Límites	Monto R D + monto solicitado	Saldo cuenta	Efectivo en cajero	Denominación	Billetes más pequeño	Status
Exitoso	**Doméstica**	**Fijo**	>=50 y <=3000	**< $3,000**	**Saldo >= Monto**	**>= monto**	**$50**	**Suficiente**	
	OK	OK	$50	$0	$120	OK	OK	OK	Listo
	OK	OK	$200	$0	$550	OK	OK	OK	Listo
	OK	OK	$500	$0	$500	OK	OK	OK	Listo
	OK	OK	$1,500	$0	$1,500	OK	OK	OK	Listo
	OK	OK	$50	$1,000	$75	OK	OK	OK	Listo
	OK	OK	$200	$1,000	$450	OK	OK	OK	Listo
	OK	OK	$500	$1,000	$600	OK	OK	OK	Listo
	OK	OK	$1,500	$1,000	$1,600	OK	OK	OK	Listo
No exitoso	**Doméstica**	**Fijo**	>=50 y <=3000	**< $3,000**	**Saldo >= Monto**	**>= monto**	**$50**	**Insuficiente**	
	OK	OK	$50	$0	$1,000	OK	OK	OK	Listo
Exitoso	**Doméstica**	**Fijo**	>=50 y <=3000	**< $3,000**	**Saldo >= Monto**	**>= monto**	**$100**	**Suficiente**	
	OK	OK	$200	$0	$234	OK	OK	OK	Listo
	OK	OK	$500	$0	$785	OK	OK	OK	Listo

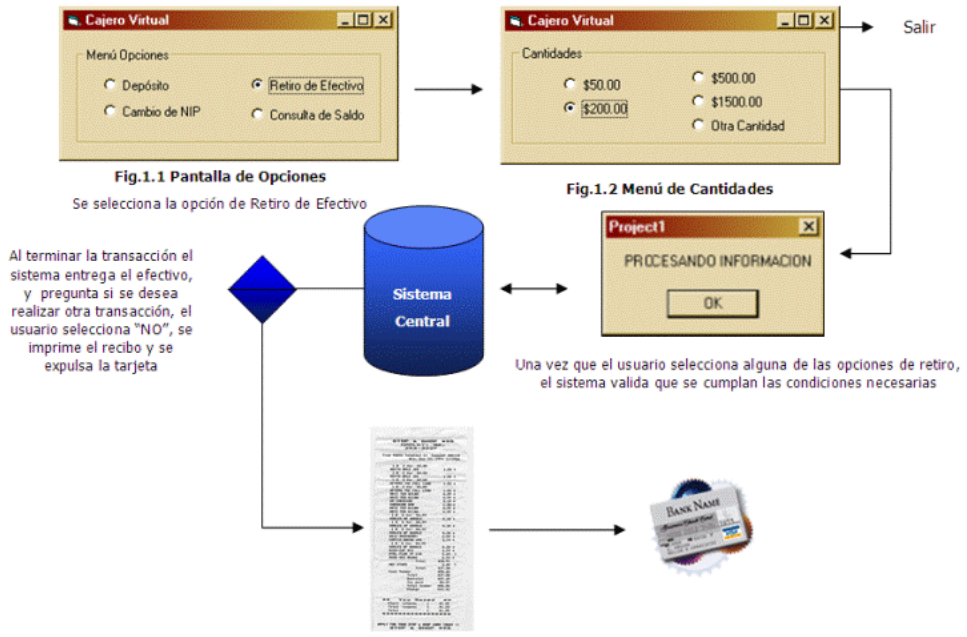
Exitoso / no exitoso	Tipo de tarjeta	Tipo de monto	Límites	Monto R D + monto solicitado	Saldo cuenta	Efectivo en cajero	Denominación	Billetes más pequeño	Status
	OK	OK	$1,500	$0	$14,622	OK	OK	OK	Listo
	OK	OK	$200	$1,000	$300	OK	OK	OK	Listo
	OK	OK	$500	$1,000	$2,300	OK	OK	OK	Listo
	OK	OK	$1,500	$1,000	$1,550	OK	OK	OK	Listo
No exitoso	**Doméstica**	**Fijo**	>=50 y <=3000	**< $3,000**	**Saldo >= Monto**	**>= monto**	**$100**	**Insuficiente**	
	OK	OK	$100	$0	$2,000	OK	OK	OK	Listo
Exitoso	**Doméstica**	**Fijo**	>=50 y <=3000	**< $3,000**	**Saldo >= Monto**	**>= monto**	**$200**	**Suficiente**	
	OK	OK	$200	0	250	OK	OK	OK	Listo
	OK	OK	$200	1000	350	OK	OK	OK	Listo
No exitoso	**Doméstica**	**Fijo**	>=50 y <=3000	**< $3,000**	**Saldo >= Monto**	**>= monto**	**$200**	**Insuficiente**	
	OK	OK	200	0	300	OK	OK	OK	Listo
Exitoso	**Doméstica**	**Fijo**	>=50 y <=3000	**< $3,000**	**Saldo >= Monto**	**>= monto**	**$500**	**Suficiente**	
	OK	OK	500	0	600	OK	OK	OK	Listo
									...

- **Mapas de pruebas:** Un mapa de pruebas también conocido como mapas de transición de estados o grafos, son una representación gráfica de los requerimientos funcionales a probar. Estos mapas son recomendables para sistemas y/o módulo complejos.

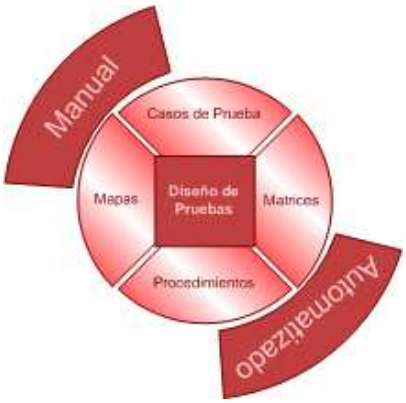
  De esta manera se tendrá una vista general que nos ayude a entender la manera como trabajará la aplicación.



También se recomienda el uso de la interfaz para representar este mapa de pruebas:



- **Scripts de pruebas:** Un script de prueba es un conjunto de instrucciones entendibles por la computadora que automatizan la ejecución de un método de prueba (o de la porción de un método de prueba) y se usan para implementar y ejecutar un caso de pruebas de forma eficiente y efectiva. Dentro del diseño de pruebas es necesario realizara un *análisis de pruebas – (¿Qué hacer manual? vs. ¿Qué automatizar?).* Ver: apéndice II - Automatización de pruebas.



Entregables (registros generados)

- Procedimientos de pruebas
- Casos de pruebas
- Matrices de pruebas
- Mapas de pruebas
- Script de pruebas

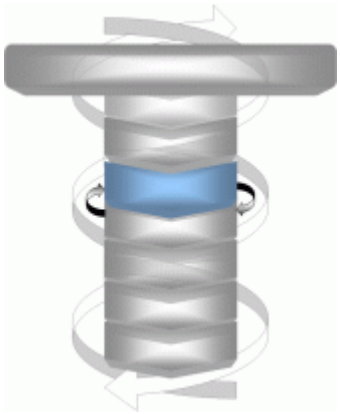
5.3.4. Ejecución de pruebas

Objetivo general
Identificar y comunicar la información necesaria sobre como ejecutar los casos de prueba de forma apropiada y consistente.

Propósito:

- Entregar métricas respecto al las pruebas realizadas.
- Revisar los resultados de la ejecución de pruebas.
- Organizar e Investigar los resultados para su evaluación.

Introducción



Durante las fases previas a la ejecución se ha hablado mucho sobre el análisis y el diseño del testware basado en las especificaciones funcionales, pero es durante esta fase donde finalmente se aplicarán los artefactos de prueba previamente diseñados y construidos a la aplicación para la detección y corrección de errores.

"Probar es el proceso de identificar defectos, cuando un defecto es cualquier variante entre lo que existe y lo que se esperaba".

Es importante conocer que la ejecución de pruebas se puede hacer de forma manual o con ayuda de herramientas (automatización).

Proceso, ¿Qué debo de hacer?
Las actividades llevadas a cabo durante la ejecución de pruebas son responsabilidad principalmente de los ingenieros de pruebas, usuarios y analistas de negocio.

Las actividades principales que deben de ser realizadas son:

- **Ejecución.** El responsable de pruebas debe ejecutar todo el testware que se elaboró durante la fase de diseño, de esta forma irá encontrando y reportando errores. Es necesario llevar un control del estatus (exitoso, no exitoso y sin probar) de cada uno de los artefactos de pruebas (principalmente casos de prueba). De esta manera se tendrá visibilidad de la cobertura de pruebas.

- **Evaluación de pruebas.** Una vez que el responsable termine un ciclo de pruebas completo se deberá de aplicar la "paradoja del pesticida", la cuál se detalla más adelante en este documento.

- **Reportes:** Se deben ir elaborando reportes de forma periódica sobre la cobertura de pruebas, la cantidad de errores reportados y sus estatus, prioridades de éstos, numero de casos de prueba ejecutados, etc.

En las siguientes secciones se describirá la manera de ejecutar las pruebas. Es importante mencionar que dentro de la *fase de diseño* se define como se realizarán las pruebas.

Aplicación de técnicas y tipos de pruebas

En la ejecución de pruebas es cuando se deben de aplicar las técnicas y tipos de pruebas, dentro de cada una de las etapas en las que se encuentre la aplicación que se esta probando.

Métodos Manuales

- **Pruebas de escritorio (verificación individual):** Esta verificación se realiza cuando se tiene el modelo de la aplicación y se llevan a cabo las pruebas en papel, es decir que se hace una comprobación utilizando varios datos que permitan establecer el correcto funcionamiento del modelo sin el uso del sistema.

- **Inspecciones (checklist):** Es un hecho que la mayoría o el grueso de los errores dentro de la aplicación se deben detectar al aplicar los casos de prueba pero también es una buena practica tomar el control de errores y a manera de checklist tratar de reproducirlos una vez mas y verificar que ninguno de éstos siga existiendo en el proyecto.

- **Verificación de requerimientos:** Uno de los objetivos principales durante la ejecución de pruebas es el asegurar que todos los requerimientos estén cubiertos y funcionen de la manera correcta dentro de la aplicación. Esto se irá verificando mientras se ejecuta el testware basado en las especificaciones funcionales previamente construido en la fase de diseño.

- **Revisiones formales (verificación y validación):** Este tipo de pruebas manuales se refiere al momento de ejecutar formalmente todo el diseño de pruebas (casos, procedimientos, mapas y matrices de pruebas) y llevar un control de los resultados obtenidos en cada uno de los ciclos de pruebas. Cabe mencionar que es la manera más recomendable de ejecutar pruebas manuales debido a que existe un diseño que soporta y valida la manera como estamos realizando las pruebas.

- **Error guessing:** Una vez que se va adquiriendo experiencia en la ejecución de pruebas, se van conociendo las combinaciones que tienen mas probabilidad de generar un error, sin importar el tipo de software que se este probando. Cuando se esta verificando la funcionalidad de un sistema se pueden detectar defectos al aplicar estos conocimientos, los cuales podrían no estar documentados en el testware como un caso de prueba pero que se tienen que registrar durante la fase de estabilización y documentación.

Automatización de pruebas

Dentro de la guía de la Fase de Diseño se marcan las ventajas y características tanto de diseñar como de ejecutar las pruebas de esta manera. Básicamente en esta fase es en donde se podrían en práctica las rutinas (scripts) y herramientas que nos ayudarán a que los procesos de pruebas sean más eficientes.

Paradoja del pesticida (evaluación de pruebas)

Cuando tenemos un problema de insectos en casa, es probable que el pesticida que estamos utilizando no este eliminando todos los insectos o que estos a lo largo del tiempo se vuelvan inmunes.

Haciendo alusión a lo anterior, dentro de las pruebas de software es recomendable que al terminar cada ciclo de pruebas, nuestro diseño de pruebas y/o la manera como las estamos ejecutando sean evaluadas para revisar y agregar más casos de prueba que es posible que estemos omitiendo o que al modificarlos (mejorarlos) podamos encontrar más errores.

Esto lo puede realizar el líder de pruebas responsable del proyecto, aunque la recomendación es que otro profesional de pruebas realice esta evaluación.

Datos de pruebas

Los datos que se utilicen para las pruebas deben de ser controlados para poder verificar la correcta integridad de la información. Es importante tomar en cuenta que esto aplica tanto para pruebas manuales como automatizadas.

- **Datos reales:** Durante la validación de operaciones del sistema o cuando éste tenga que hacer uso de datos, se deben de ocupar muchos ejemplos reales, es decir los mismos datos que la aplicación tendrá que soportar cuando se encuentre productiva, esta es una forma de asegurar que las operaciones y resultados serán congruentes una vez que el sistema sea liberado.

- **Controlar los datos que se están utilizando:** Al estar validando la aplicación se debe controlar los datos de prueba que se usan como entrada para que finalmente cuando se ejecuten los casos de prueba y se obtengan resultados, éstos sean consistentes

- **Utilizar subtécnicas de caja negra:** Para la comprobación de resultados usando datos de prueba se recomienda usar subtécnicas de caja negra como test to pass, test to fail, condiciones límite, partición equivalente, etc. El detalle de estas subtécnicas se encuentra descrito en el capitulo 1 “Fundamentos teóricos”.

- **Analizar las entradas y salidas:** Conocer cuales son los medios de entrada de información (teclado, mouse, USB drives, archivos de cargas masivas, etc.) que podría utilizar la aplicación en cualquier momento y cuales serían los resultados esperados ó su forma de salida (impresión, envío de un archivo, mail, etc.). Así mismo tener un control sobre los resultados esperados y los resultados obtenidos, cualquier diferencia entre estos debe ser considerado un error.

- **Ejecutar ataques diseñados para sobrepasar los límites de entradas y salidas:** Se pueden realizar combinaciones que normalmente el sistema no esperaría o que regularmente no sucederían en un ambiente productivo pero sin embargo podrían dar como resultado un error.

- **Analizar la manera en que el software almacena y manipula los datos:** De esta manera se podrá verificar que los procesos que realiza la aplicación están funcionando correctamente. Aquí es importante revisar el repositorio de información (base de datos) que utiliza el sistema.

- **Ejecutar ataques diseñados para corromper los datos:** Tratar de modificar información cuyo propósito esta bien definido, por ejemplo tratar de borrar, editar, copiar o cortar datos que son exclusivos de consulta.

- **Analizar como el software realiza los cálculos:** Revisar todos los procesos, operaciones o transacciones que puedan involucrar un cálculo y que se espere un resultado especifico de él.

- **Ejecutar ataques para forzar errores en el cálculo:** Para aquellas operaciones que involucren un cálculo se debe de comprobar cual sería la respuesta al ingresar datos erróneos como números negativos, decimales largos, caracteres especiales, etc.

- **Conocer el ecosistema en el que reside sistema:** Conocer como es la convivencia que tiene la aplicación en el ambiente en el que reside, es decir su interacción con el sistema operativo, los procesos que utiliza para ejecutar acciones, las veces que invoca dichos procesos, las dll’s que utiliza para funcionar para así al encontrar un error se pueda dar mas información al equipo de desarrollo sobre donde esta la falla y agilizar la corrección de errores.

Entregables (registros generados)

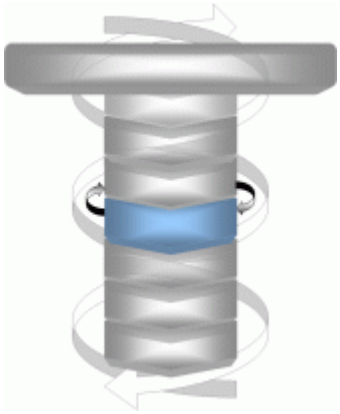
- Reportes de pruebas.
- Archivos de datos.
- Testware actualizado (cobertura de pruebas, estatus de casos de prueba y mantenimiento del control de errores).

5.3.5. Control y seguimiento de errores

Propósito

- Capturar los defectos en una herramienta para dar lugar al proceso de corrección.
- Control el seguimiento y administración de defectos.

Introducción



Dentro del proceso se ha hablado mucho con respecto cada unas de las actividades de las pruebas de software pero ¿Qué pasa cuando encontramos un defecto?, ¿Cómo y dónde lo tenemos que documentar?, ¿Cuál es ciclo de vida que tiene un defecto?, etc.

Con base a lo anterior es indispensable contar con un control y seguimiento de errores correcto teniendo como objetivo: asegurar que los productos de software construidos no sean liberados con defectos o errores teniendo como base los requerimientos establecidos por el cliente en la especificación de requerimientos.

Es importante comenzar con definir lo que es un defecto (error o bug): **"Manifestación de un error en el software"**.

Características:

- Son la razón principal (objetivo) de las pruebas de software y la meta es encontrarlos antes que el usuario.
- Corregir los errores para fortalecer el producto y proveer de una manera detallada una descripción del problema y como fue encontrado.
- Sinónimos: fault, defect, problem, error, incident, anomaly, variance, failure, pulgas, etc.

Proceso, ¿Qué debo de hacer?

Primero es importante comentar que esta actividad es responsabilidad de líderes de proyectos, programadores e ingenieros de pruebas.

Para llevar un correcto control y seguimiento de errores es necesario contar con control de errores debido a que será la manera como se registrarán las deficiencias que han sido detectadas en el sistema. De esta manera los programadores tendrán la información necesaria para corregir estos problemas.

Este control será donde se llevará una relación precisa de cada uno de los errores encontrados así como su seguimiento para su pronta corrección.

Las actividades principales que deben de ser realizadas son:

Responsable	Actividad
Usuario, BA, STE	**Pruebas.** Al terminar el proveedor cierta funcionalidad el ingeniero de pruebas debe de aplicar las pruebas necesarias basándose principalmente en las en su diseño de pruebas correspondiente a la funcionalidad(es) o caso de uso(s) en cuestión.
Usuario, BA, STE	**Documentación de los errores encontrados.** Cada que encuentre un error o defecto contra los requerimientos establecidos y/o diseño de pruebas, el responsable de pruebas deberá de registrar el error en alguna herramienta de seguimiento de errores. La información que deberá de registrar en este momento es la siguiente: 1. Fecha 2. Versión 3. Prioridad 4. Tipo de error 5. Pasos para obtenerlo 6. Localización 7. Estatus 8. Encontró 9. Generó 10. Solucionó 11. Fecha de corrección 12. Comentarios
Usuario, BA, STE	**Informar del error.** El responsable de pruebas debe actualizar en la herramienta de seguimiento de errores, los nuevos errores encontrados para que lo puedan consultar los programadores y el BSA. La recomendación es utilizar este tipo de seguimiento en todos los proyectos.
Proveedor, desarrollador	**Consulta del error.** Los programadores deben revisar continuamente la lista de errores encontrados para ver si tienen errores por corregir.
Proveedor, desarrollador	**Corrección del error.** Cada programador debe de ir corrigiendo sus errores tomando en consideración la prioridad asignada y la dependencia que ocasionen los errores para seguir avanzando en su desarrollo. Cuando corrija algún error debe de cambiar el estatus del error a “corregido”. También podrá escribir algunas observaciones si así lo requiere.
BSA, BA, QA y proveedor o desarrollador	**Cancelación del error.** En caso de que el programador encuentre que un error registrado en el control de errores no sea realmente un error deberá de notificárselo al responsable de pruebas para asegurarse de que realmente no sea un error consultando la especificación de requerimientos, al líder de proyecto para que lo autorice. El responsable de pruebas debe cambiar el estatus a “cancelado” de la lista. En caso de cancelarlo debe registrar quién recomendó o autorizó la cancelación del error. El de marcarlo como “cancelado”, es con el fin de que si llegara a haber alguna queja respecto a este error se pueda identificar quien autorizó que no se corrigiera.
Usuario, BA, STE	**Validación de la corrección del error.** El responsable de pruebas debe de revisar continuamente la lista de errores para ver cuáles son los errores que han sido corregidos por los programadores, pero que no han sido validados. El responsable de pruebas debe de probarlos para asegurarse de que realmente sea así. En caso de que compruebe que ya fue corregido el error deberá registrarlo como “revisado”, en caso contrario deberá de quitarle el estatus de “corregido” y de preferencia avisarle al programador para que este se asegure nuevamente de corregirlo y de marcarlo con dicho estatus.

Responsable	Actividad
BSA	**Seguimiento de errores.** El BSA ó responsable del proyecto debe de revisar continuamente la lista o control de errores para garantizar que los errores se vayan corrigiendo con suficiente frecuencia.
QA, BSA	**Fin del proyecto.** Al liberar el sistema, el ingeniero de pruebas o el líder de proyecto, deberá de llenar un Informe final de pruebas que indique si existen errores sin corregir y las razones de que sea liberado así el sistema, este informe es para control interno. Además, deberá de ser firmado por el líder de proyecto, pues es el único autorizado y responsable para liberar el producto con errores.

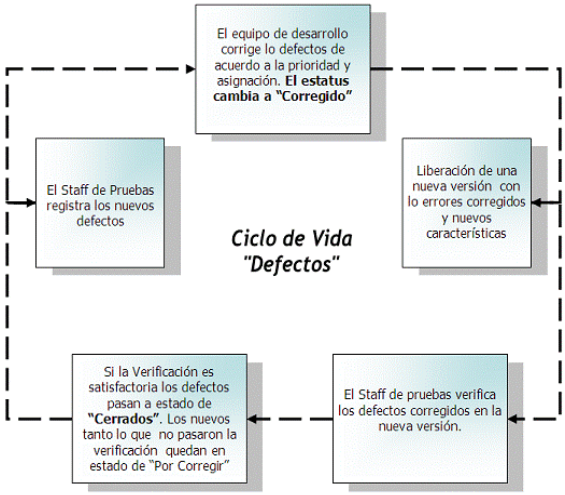
En ocasiones es posible que el producto se tenga que liberar con errores, debido a circunstancias como la presión del tiempo. Los errores registrados en esta lista con prioridad de "S1" deberán ser corregidos y validados por el BSA. En cuanto sea posible se tratará de corregir los demás errores que hubieran quedado en esta lista.

Elementos de un control de errores
La lista de severidades se muestra a continuación:

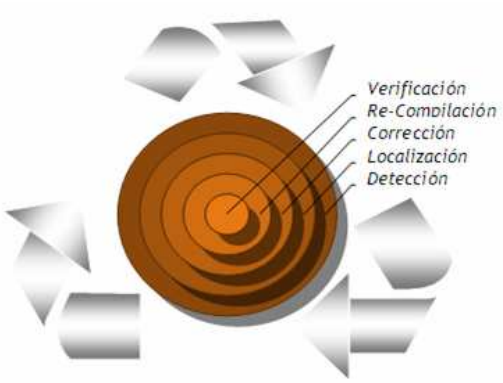
1. Severidad:

   - S1 Alta: La aplicación NO esta disponible. Se detiene el proceso de operación. No se puede continuar. Requiere atención INMEDIATA del desarrollador.
   - S2 Media: No está operando una función importante y las actividades están seriamente obstaculizadas.
   - S3 Baja: Problemas menores, la función opera, pero el resultado NO es el esperado, las actividades NO se encuentran obstaculizadas.
   - S4 Cambio de requerimiento: Sugerencias / recomendaciones, relacionadas con el uso de términos / ortografía / gramática / formato, impresión, etc.

Ciclo de vida de los defectos
Anteriormente se mencionaron cada una de las actividades del seguimiento de errores, a continuación a manera de resumen se presenta el "ciclo de vida de un defecto":



Es importante comentar que este ciclo de vida será un proceso repetitivo durante la ejecución del proceso de pruebas como lo muestra la siguiente figura:



¿Los errores son casos de prueba?
La respuesta es SI, los errores encontrados son parte del testware y al ser documentados pueden utilizarse como casos de prueba. En caso de que no se pueda entonces quiere decir que el error no puede ser reproducido y por lo tanto no puede ser corregido.

Algo muy importante es que al ser parte del inventario de pruebas y serán usados en las pruebas de regresión.

Con base a lo anterior al ejecutar cada uno de los ciclos de pruebas (posteriores al primero) se deberá de contemplar la lista de errores como parte del diseño de pruebas a ejecutar.

Entregables (registros generados)

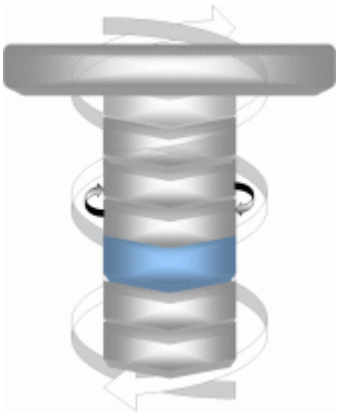
- Control de errores del proyecto.

5.3.6. Estabilización y documentación

Objetivo general

Verificar los últimos detalles de la aplicación, tanto respecto a la documentación del testware generado como ejecutar pruebas de regresión y aceptación.

Introducción



Una vez terminada la ejecución de pruebas y la corrección de errores, la aplicación estará lista para pasarla a los usuarios finales pero antes de llegar a este punto se debe de asegurar que la versión del sistema mantiene su integridad, es decir que siga funcionando de la misma forma que durante la ejecución de pruebas.

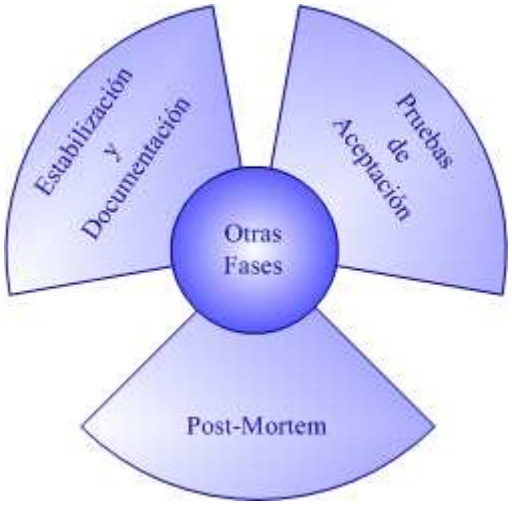
Durante esta fase se deben ejecutar pruebas ya que algunas veces se aplica un “tunning” a la aplicación (por ejemplo para mejorar el performance) o cuando se hacen cambios de último momento se debe verificar que las funciones de negocio no hayan sufrido cambios o hayan sido afectadas,

para esto se debe correr el “happy path”, es decir se tienen que correr un ciclo de pruebas de regresión a la aplicación.

Al terminar esta estabilización sería el momento de liberar la aplicación, módulo ó caso de uso probado para su validación (recordar que parte de lo que marca el proceso es no esperar hasta que se tenga todo el sistema desarrollado para validar con los usuarios y/o clientes).

Finalmente al terminar de ejecutar todo el proceso de pruebas se deberá de realizar un análisis de lo que se hizo bien y lo que falló para así dar lugar a la mejora continua. De esta manera la forma como se realizan las pruebas se seguirá robusteciéndose buscando ser más efectivos y aumentando la calidad de las aplicaciones.

“La calidad de un sistema está estrechamente relacionada a la calidad con que se realizan las pruebas del mismo."



Proceso, ¿Qué debo de hacer?

Para validar la integración de la aplicación se deben de ejecutar los casos de pruebas más representativos de la funcionalidad y de negocio.

Responsable	Actividad
STE, usuario, BA	**Pruebas.** Al terminar de ejecutar todo el testware y verificar que los errores hayan sido corregidos, se hace una liberación de la versión de la aplicación por parte del proceso de pruebas.
STE, usuario, BA	**Selección de casos de prueba.** El responsable de pruebas debe hacer una selección de casos de prueba que se encuentren dentro de su testware y que representen a grandes rasgos la funcionalidad básica y la de negocio. También se debe de tomar una muestra de los errores registrados y a forma de check list verificar que éstos no hayan regresado a la aplicación.
STE, usuario, BA	**Ejecución de casos de prueba.** Se deben ejecutar los casos de prueba previamente seleccionados con el fin de asegurar que la versión a liberarse a pruebas con usuarios mantiene su integridad.

Responsable	Actividad
STE, usuario, BA	**Liberación a usuarios.** Una vez concluidas las pruebas de estabilización y que todo este funcionando correctamente, la versión está lista para enviarla a los usuarios finales para su Vo.Bo. En caso de que se encuentre algún error, se debe de reportar y entraría en el ciclo de corrección y seguimiento.
QA, STE	**Documentación del testware.** Durante esta fase los miembros del equipo de pruebas deben revisar que el testware este completo, además debe utilizarse este tiempo para documentar o actualizar todos aquellos artefactos que hayan cambiado.

Nota. Los errores encontrados por los usuarios se deben de registrar para llevar la cuenta de las cosas que se no se encontraron durante la fase de ejecución de pruebas por el equipo de testing.

Entregables (registros generados)

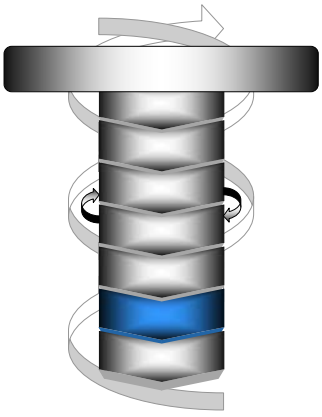
- Testware actualizado y consistente.

5.3.7. Pruebas de aceptación

Objetivo general

Garantizar la satisfacción total del usuario final del producto entregado, así como la aceptación de la documentación del testware generado durante el proyecto.

Introducción



Durante las fases previas a la ejecución se ha hablado mucho sobre el análisis y el diseño del testware basado en las especificaciones funcionales, pero es durante esta fase donde finalmente se aplicarán los artefactos de prueba previamente diseñados y construidos a la aplicación para la detección y corrección de errores.

"Probar es el proceso de identificar defectos, cuando un defecto es cualquier variante entre lo que existe y lo que se esperaba"

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.

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 comienza a usarlo.

Es indispensable sensibilizar a los usuarios de la importancia de está fase así como informar con tiempo periodo en que se realizarán dichas pruebas.

Proceso, ¿Qué debo de hacer?

Las pruebas de aceptación son aquellas que son realizadas por el usuario final, quien compara la aplicación con los requerimientos originales.

Se recomienda que las validaciones con usuarios se vayan haciendo en las partes de la aplicación que se vayan liberando y no solo cuando se tenga todo el producto completo.

Mas detalle sobre las pruebas de aceptación se puede encontrar en la sección de etapas, técnicas y tipos de pruebas (ver capítulo 1).

Entregables (registros generados)

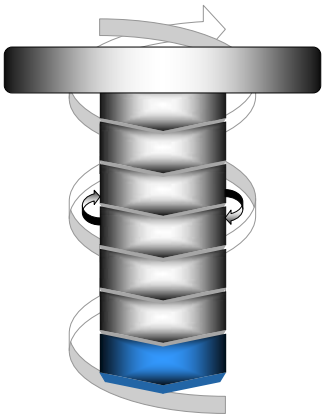
- Acta de pruebas de aceptación.

5.3.8. Post mortem

Objetivo general

Cubrir los procesos que requieran un ajuste, destacando los procesos más efectivos, así como las peores practicas (áreas de oportunidad) para tener una mejora continua.

Introducción



El post mortem se refiere a la revisión de las actividades o aspectos dentro del proyecto que se llevaron a cabo correctamente, así como los que necesitan mejorar.

"Al final de la reunión se cuenta con una lista de cosas, las cuales hay que priorizar y planear las acciones para mejorarlas"

Una fase de post mortem tiene como objetivo el "mirar hacia atrás" y revisar las cosas que se hicieron bien y las cosas que necesitan mejorar. Las revisiones de post mortem se encargan de cubrir los procesos que necesitan algún ajuste, así como destacar los procesos más efectivos y así dar como resultado las mejores prácticas para mejorar y robustecer el diseño y ejecución de pruebas para futuros proyectos.

Proceso, ¿Qué debo de hacer?

Las revisiones post mortem se realizan después de que el sistema se encuentra en el ambiente de producción y se debe de citar al menos una semana antes para que todos tengan el tiempo de reflexionar sobre los problemas o issues del proyecto a los que se enfrentaron. Se debe de pedir a los asistentes que lleguen a la reunión con lo siguiente:

- Una lista con dos o más cosas que consideran se hicieron bien.
- Una lista con dos o más cosas que creen que se hicieron mal.
- Sugerencias para mejorar las prácticas en futuros proyectos.

Durante la reunión se tiene que recavar la información mencionada antes y a medida que cada persona de su opinión, se debe de ir categorizando y así ir contabilizando los comentarios que son parecidos. Lo anterior permite contemplar cuantas personas tienen las mismas observaciones sobre lo sucedido en el proyecto.

Al final de la reunión se contará con una lista de las cosas que se mencionaron frecuentemente, dicha lista debe de revisarse y hacer una priorización de los elementos que contiene para distinguir cuales son los mas importantes. Finalmente se debe elaborar una lista de acciones a realizar para mejorar aquellas observaciones y/o comentarios para posteriormente publicar el resultado.

Al comienzo de los siguientes proyectos, los miembros del equipo deben de revisar el último reporte post mortem para implementar las acciones propuestas en él. Se recomienda que el responsable de controlar el reporte de post mortem sea el líder o responsable de pruebas.

Entregables (registros generados)

- Reporte post mortem.

5.4. INSTRUCTIVOS DE FORMATOS

Esta sección del documento describe el uso de cada formato ajustado al proceso implantado en la organización.

5.4.1. Administración de pruebas

Bitácora de pruebas

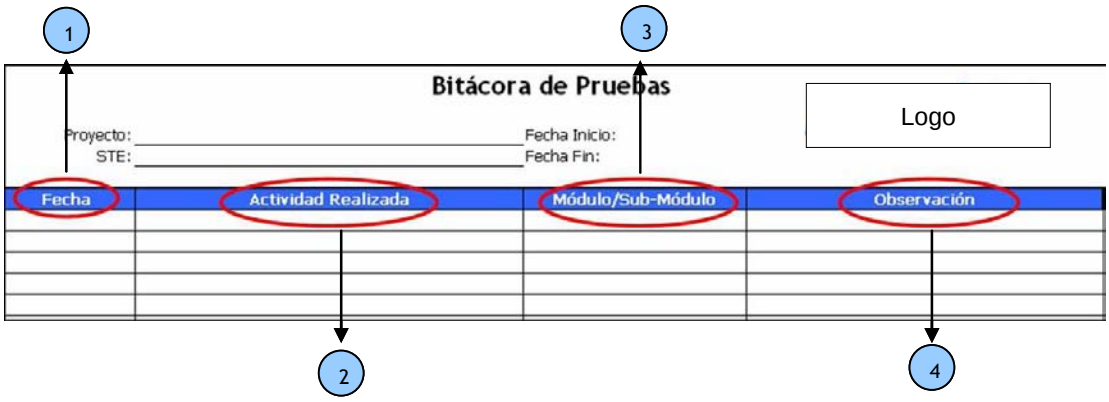
Formato diseñado para el registro de todas las actividades llevadas a cabo por cada miembro del equipo durante el desarrollo de un proyecto.

Datos de Identificación:

- Proyecto
- Ingeniero de pruebas de software (STE - Software Test Engineer)
- Fecha inicio
- Fecha fin

Campos:

Nombre del campo	Descripción	Número
Fecha	Fecha de realización de la actividad que se esta registrando.	1
Actividad realizada	Descripción de la actividad que se lleva a cabo.	2
Módulo / submódulo	Nombre del módulo, submódulo, caso de uso o requerimiento funcional sobre el cual la actividad registrada esta relacionada.	3
Observación	Campo para agregar cualquier observación, comentario o información adicional sobre la actividad.	4



Control de riesgos

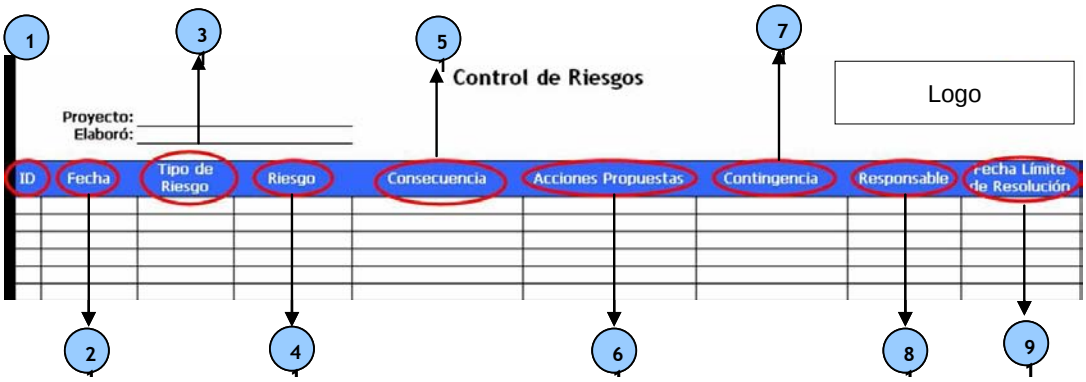
Formato diseñado para el registro de todos aquellos riesgos que hayan sido identificados, analizados y se tengan que escalar en cualquier punto de un proyecto.

Datos de Identificación:

- Proyecto
- Elaboró

Campos:

Nombre del campo	Descripción	Número
ID	Identificador único con el que se puedan localizar fácilmente los riesgos.	1
Fecha	Fecha en la cual se encontró o identificó el riesgo.	2
Tipo de riesgo	Categorización de los riesgos, dividiéndolos en riesgos en la gente, en tecnología o en la organización.	3
Riesgo	Una descripción detallada de éste.	4
Consecuencia	Descripción del problema en que el riesgo se podría convertir en caso de no mitigarlo.	5
Acciones propuestas	Dar, siempre que sea posible, sugerencias sobre como mitigar el riesgo.	6
Contingencia	Plan de que hacer en caso de que no se pueda lidiar con el riesgo o que éste ya se haya convertido en un problema.	7
Responsable	Nombre de la persona más indicada para resolver o tomar responsabilidad por el riesgo reportado.	8
Fecha límite de resolución	Fecha máxima en la cual se tiene que dar una solución al riesgo.	9



Control de pruebas

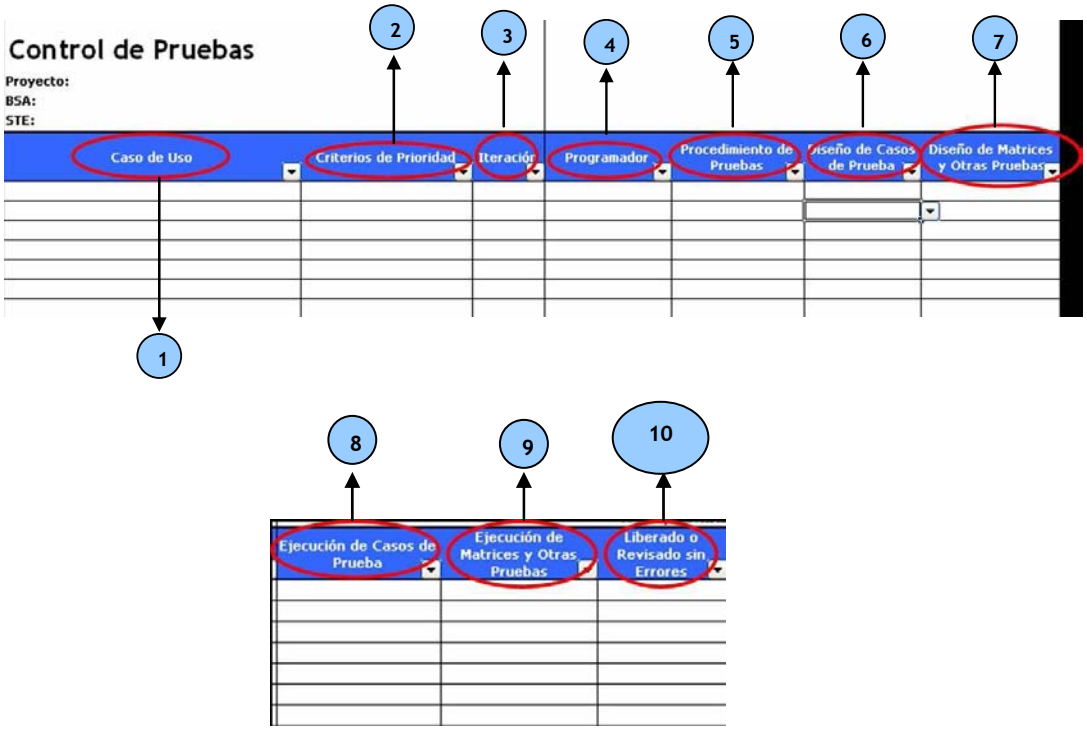
Durante las diferentes etapas del proceso se ven generando entregables, los cuales sirven tanto para realizar las pruebas como para generar los diferentes reportes necesarios para identificar el avance que se lleva.

Datos de Identificación:

- Proyecto
- Analista de sistemas de negocio (BSA - Business System Analyst)
- Ingeniero de pruebas de software (STE - Software Test Engineer)

Campos:

Nombre del campo	Descripción	Número
Casos de uso	Nombre del caso de uso o requerimiento funcional.	1
Criterios de prioridad	Selección de la prioridad que tiene un requerimiento funcional.	2
Iteración	Número de iteración o ciclo de prueba en el que se encuentra el requerimiento funcional.	3
Programador	Nombre del programador asignado a desarrollar y corregir errores del requerimiento funcional.	4
Procedimientos de pruebas	Estatus en el que se encuentra el diseño de procedimientos de prueba para el requerimiento funcional.	5
Diseño de casos de prueba	Estatus en el que se encuentra el diseño de casos de prueba para el requerimiento funcional.	6
Diseño de matrices y otras pruebas	Estatus en el que se encuentra el diseño de matrices de prueba para el requerimiento funcional.	7
Ejecución de casos de casos de prueba	Estatus en el que se encuentra la ejecución de procedimientos de prueba para el requerimiento funcional.	8
Ejecución de matrices y otras pruebas	Estatus en el que se encuentra la ejecución de casos de prueba para el requerimiento funcional.	9
Liberado o revisado sin errores	Estatus general del requerimiento funcional.	10



Reporte y control de versiones

Un control de versiones es un registro donde se tienen documentadas las características de cada versión, release o build liberado por el área de desarrollo. A cada versión se le ejecutan los diferentes artefactos de prueba y se van encontrando diferentes errores.

Datos de Identificación:

- Fecha
- Proyecto

Campos:

Nombre del campo	Descripción
No. de versión / build / release	Número con el que se identifica la versión que está en proceso de pruebas.
Fecha de liberación por desarrollo	Fecha en la cual el equipo de desarrollo terminó de hacer las pruebas unitarias y se encuentra en condiciones de pasar a la ejecución de pruebas.
Fecha de liberación por pruebas	Fecha en la cual el equipo de pruebas terminó de ejecutar su testware y está lista para pasar a pruebas de aceptación.
Funcionalidad agregada	Lista de la funcionalidad agregada a la versión.
Funcionalidad no liberada	Lista de funcionalidad faltante.
No. de errores encontrados	Número de errores nuevos.
No. de errores por corregir	Número de errores que todavía existen en la versión.

Nombre del campo	Descripción
No. de errores cancelados	Número de errores que por algún motivo fueron cancelados.
No. de errores cerrados	Número de errores corregidos y que han pasado la verificación de calidad.
Estatus	Dictamen de la versión, ¿Debe ser liberada o no?
Comentarios	Campo para agregar algún comentario u observación sobre la versión registrada.

5.4.2. Análisis y planeación de pruebas

Especificación de pruebas

Formato diseñado para plasmar todos los requerimientos funcionales y no funcionales de la aplicación, la estrategia de pruebas, recursos necesarios, información general del proyecto, etc.

Campos:

Elemento	Descripción corta
Introducción • Propósito • Antecedentes • Alcance • Identificación del proyecto	Resumen tanto del proyecto como de la organización. ¿Con que información cuento?
Requerimientos a Probar • Casos de uso, requerimientos funcionales • Requerimientos no funcionales	¿Qué voy a Probar?
Estrategia de pruebas • Proceso de pruebas • Etapas, técnicas y tipos de pruebas • Herramientas	¿Cómo lo voy a hacer?
Recursos • Recursos del sistema • Recursos humanos	¿Qué necesito?
Actividades del proyecto	¿Cuánto tiempo necesito y que es lo que haré?
Criterios de aceptación del sistema	¿Qué criterios deben cumplirse?, Estos serán validados por el cliente para la liberación del sistema.

Plan de pruebas

El plan de las pruebas es un formato para registrar todas las actividades, tiempos y recursos para llevar a cabo el proyecto como está definido.

Nombre	Descripción
Actividades	Detallar las actividades a realizar durante el proceso de las pruebas del proyecto.
Tiempos	Asignar los tiempos de cada una de las actividades. Definiendo fecha inicio y fin de cada una de ellas.
Recursos	Definir los recursos que estarán involucrados en las actividades y los cuales serán responsables de ejecutar cada una de ellas.

Plan de Trabajo - Pruebas de Software	3 days?	0%	Mon 11/10/04	Wed 13/10/04	
Análisis y Planeación	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Análisis documentación preliminar (ID del Proyecto)	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Diseñar Plan de Pruebas (Especificación de Pruebas)	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Diseñar Plan de Trabajo	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Configuración del Ambiente de Pruebas	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Diseño	3 days?	0%	Mon 11/10/04	Wed 13/10/04	
Diseño de Procedimientos de Pruebas	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Diseño de Casos de Prueba	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Diseño de Matrices Pruebas	1 day	0%	Mon 11/10/04	Mon 11/10/04	
Diseño de Mapas de Pruebas	1 day	0%	Tue 12/10/04	Tue 12/10/04	11
Diseño de Scripts	1 day	0%	Wed 13/10/04	Wed 13/10/04	12
Ejecución & Control de Errores	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Ejecución de Pruebas	1 day	0%	Mon 11/10/04	Mon 11/10/04	
Control y Seguimiento de Errores	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Documentación y Estabilización	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Actualización de Testware	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Pruebas de Regresión (Fine Tunning - Últimos Detalles)	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Pruebas de Aceptación	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Pruebas con Grupo de Usuarios I	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Pruebas con Grupo de Usuarios II	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Vo.Bo.	1 day?	0%	Mon 11/10/04	Mon 11/10/04	
Administración de Pruebas	1 day?	0%	Mon 11/10/04	Mon 11/10/04	

Configuración del Ambiente de Pruebas

Este artefacto ayudará a controlar y realizar las pruebas en diferentes ambientes, teniendo así un mejor manejo del hardware de su equipo, así como una mejor organización y control de versiones.

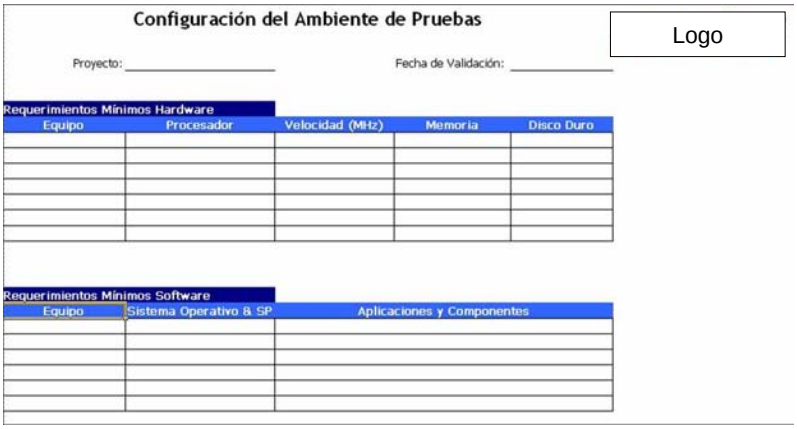
Datos de Identificación:

- Proyecto
- Fecha de Validación

Campos:

Nombre del campo	Descripción
Requerimientos mínimos de hardware	
Equipo	Nombre de la PC o equipo sobre el cual se van a ejecutar pruebas del sistema.
Procesador	Tipo de procesador que contiene el equipo de pruebas (Ej. Pentium IV)
Velocidad (Mhz)	Velocidad del procesador del equipo de pruebas

Nombre del campo	Descripción
Memoria	Memoria RAM que contiene el equipo de pruebas (Ej. 512 Mb)
Disco duro	Capacidad de almacenaje que tiene el equipo de pruebas (Ej. 60 Gb)
Requerimientos mínimos de software	
Equipo	Nombre de la PC o equipo sobre el cual se van a ejecutar pruebas del sistema.
Sistema Operativo	Sistema Operativo instalado en la PC de pruebas. Este sistema operativo debe de ser el mismo que tienen las maquinas de los usuarios finales (Ej. Windows XP Professional SP2)
Aplicaciones y componentes	Lista de todas las aplicaciones con las cuales el sistema bajo pruebas tendrá que convivir (Ej. Office 2007, Acrobat, sistema de emisión, sistema de siniestros)



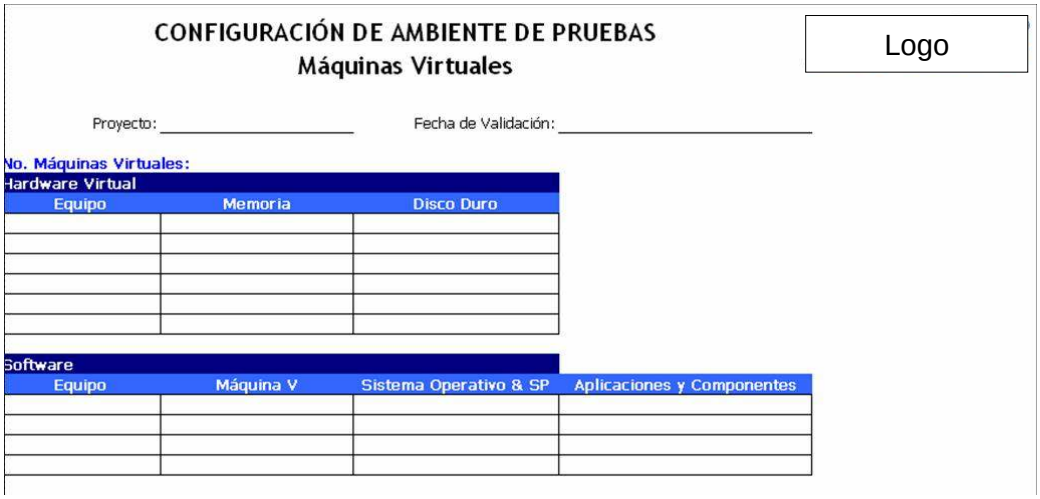
Campos (laboratorio de pruebas):

Nombre del campo	Descripción
Requerimientos mínimos de hardware	
Equipo	Nombre de la PC o equipo sobre el cual se van a ejecutar pruebas del sistema.
Procesador	Tipo de procesador que contiene el equipo de pruebas (Ej. Pentium IV)
Velocidad (Mhz)	Velocidad del procesador del equipo de pruebas
Memoria	Memoria RAM que contiene el equipo de pruebas (Ej. 512 Mb)
Disco duro	Capacidad de almacenaje que tiene el equipo de pruebas (Ej. 60 Gb)
Requerimientos mínimos de software	
Equipo	Nombre de la PC o equipo sobre el cual se van a ejecutar pruebas del sistema.
Sistema operativo	Sistema Operativo instalado en la PC de pruebas. Este sistema operativo debe de ser el mismo que tienen las maquinas de los usuarios finales (Ej. Windows XP Professional SP2)
Aplicaciones y componentes	Lista de todas las aplicaciones con las cuales el sistema bajo pruebas tendrá que convivir (Ej. Office 2003, Acrobat, sistema de emisión, sistema de siniestros)


CONFIGURACIÓN DE AMBIENTE DE PRUEBAS
Laboratorio de Pruebas
Logo
Proyecto: _____ Fecha de Validación: _____
Número de Equipos:
Hardware
Equipo | Procesador | Velocidad (MHz) | Memoria | Disco Duro
Software
Equipo | Sistema Operativo & SP | Aplicaciones y Componentes

Campos (máquinas virtuales):

Nombre del campo	Descripción
Hardware virtual	
Equipo	Nombre de la PC o equipo sobre el cual se van a ejecutar pruebas del sistema.
Memoria	Memoria RAM que contiene el equipo de pruebas (Ej. 512 Mb)
Disco Duro	Capacidad de almacenaje que tiene el equipo de pruebas (Ej. 60 Gb)
Software	
Equipo	Nombre de la PC o equipo sobre el cual se van a ejecutar pruebas del sistema.
Máquina virtual	Nombre asignado a la máquina virtual.
Sistema operativo	Sistema Operativo instalado en la PC de pruebas. Este sistema operativo debe de ser el mismo que tienen las maquinas de los usuarios finales (Ej. Windows XP Professional SP2)
Aplicaciones y componentes	Lista de todas las aplicaciones con las cuales el sistema bajo pruebas tendrá que convivir (Ej. Office 2007, acrobat, Sistema de emisión)


CONFIGURACIÓN DE AMBIENTE DE PRUEBAS
Máquinas Virtuales
Logo
Proyecto: _____ Fecha de Validación: _____
No. Máquinas Virtuales:
Hardware Virtual
Equipo | Memoria | Disco Duro
Software
Equipo | Máquina V | Sistema Operativo & SP | Aplicaciones y Componentes

5.4.3. Diseño de pruebas

Casos de prueba

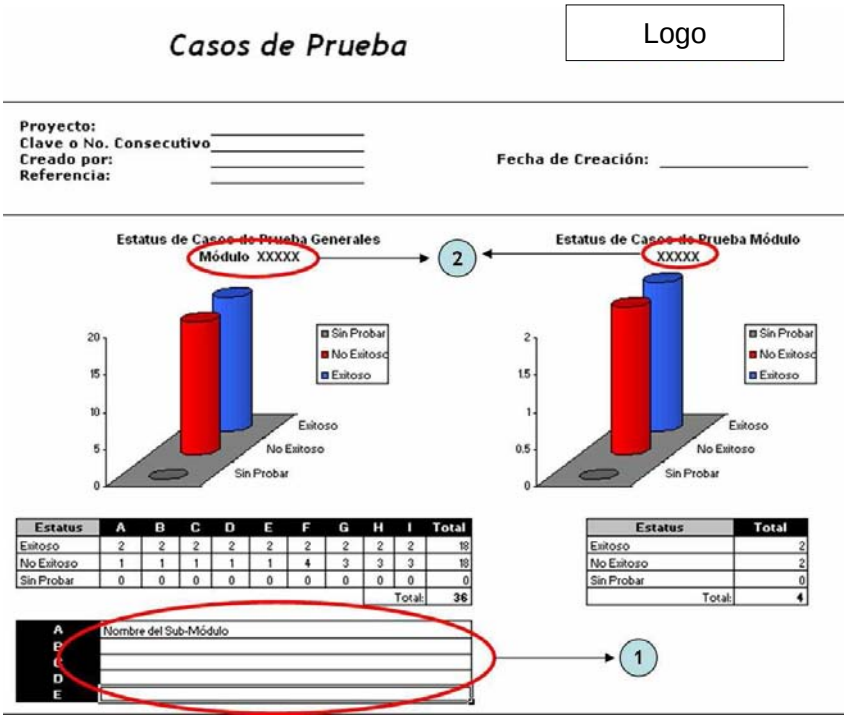
Formato diseñado para identificar y comunicar las condiciones que serán puestas en ejecución y que son necesarias para verificar y aceptar un requerimiento funcional.

Datos de Identificación:

- Proyecto
- Clave o número consecutivo
- Creado por
- Fecha de creación
- Referencia

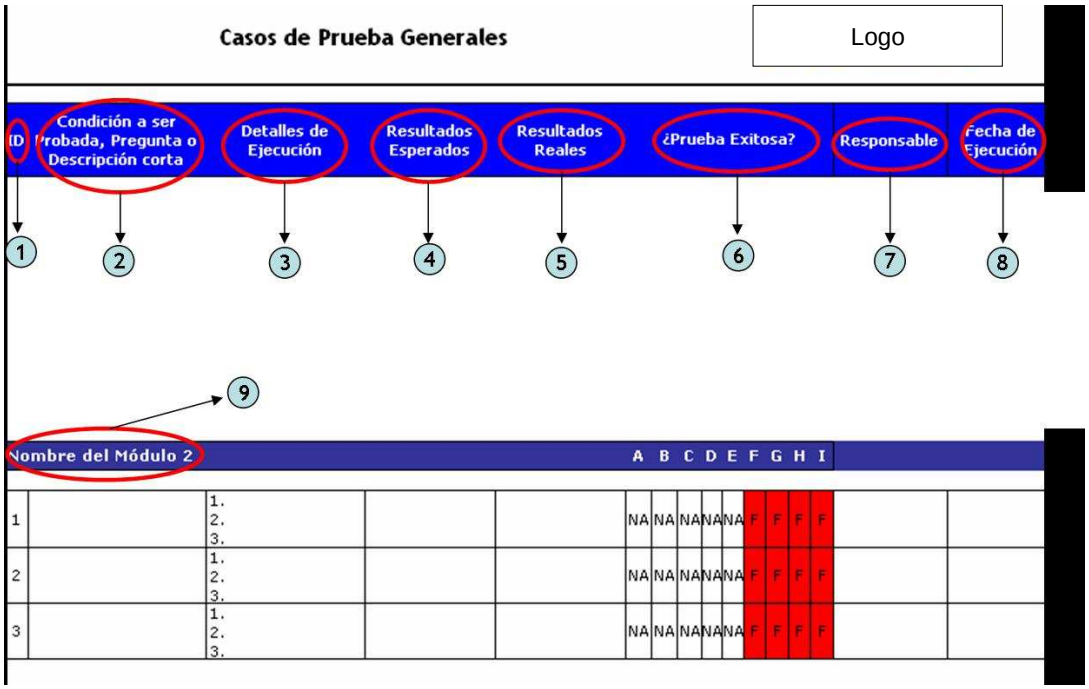
Campos (página principal):

Nombre del campo	Descripción	Número
Nombre del submódulo	Campos destinados para casos de prueba generales. Se captura el nombre de los módulos del sistema a probar en los cuales aplica el mismo caso de prueba.	1
Módulo XXXXX	Nombre del módulo(s) que se encuentren bajo pruebas.	2



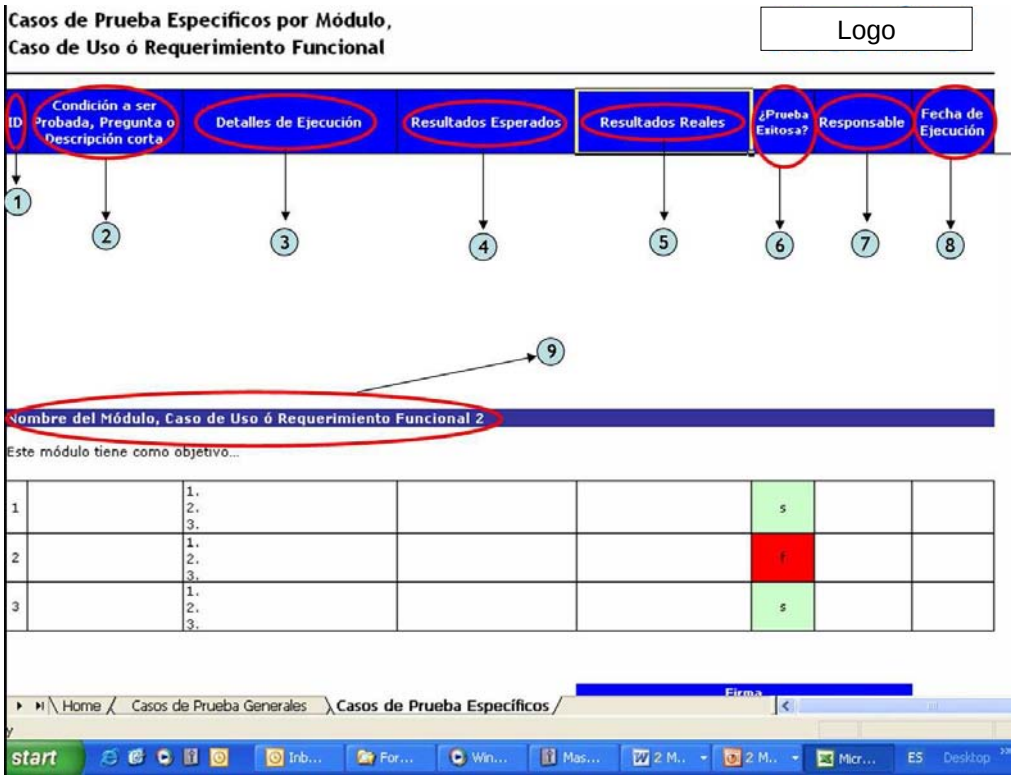
Campos (casos de prueba generales):

Nombre del campo	Descripción	Número
ID	Identificador del caso de prueba en este documento.	1
Condición a ser probada	Funcionalidad específica a ser probada (pregunta corta ó descripción corta).	2
Detalles de ejecución	Pasos a seguir para completar el caso de prueba	3
Resultados esperados	¿Qué es lo que debe ocurrir?	4
Resultados reales	¿Qué es lo que ocurrió en realidad?	5
¿Prueba exitosa?	¿La aplicación funcionó como se esperaba? (S: Sí; F: Falló, SP: Sin Probar. Si es F, se debe describir la falla en la herramienta de errores).	6
Responsable	Nombre de la persona encargada de ejecutar el caso de prueba.	7
Fecha de ejecución	Fecha en la que se ejecuta el caso de prueba.	8
Nombre del módulo	Nombre del módulo o requerimiento funcional al cual pertenecen el conjunto de casos de prueba.	9



Campos (casos de prueba específicos):

Nombre del campo	Descripción	Número
ID	Identificador del caso de prueba en este documento.	1
Condición a ser probada	Funcionalidad específica a ser probada (pregunta corta ó descripción corta).	2
Detalles de ejecución	Pasos a seguir para completar el caso de prueba.	3
Resultados esperados	¿Qué es lo que debe ocurrir?	4
Resultados reales	¿Qué es lo que ocurrió en realidad?	5
¿Prueba exitosa?	¿La aplicación funcionó como se esperaba? (S: Sí; F: Falló, SP: Sin Probar. Si es F, se debe describir la falla en la herramienta de errores).	6
Responsable	Nombre de la persona encargada de ejecutar el caso de prueba.	7
Fecha de ejecución	Fecha en la que se ejecuta el caso de prueba.	8
Nombre del módulo	Nombre del módulo o requerimiento funcional al cual pertenecen el conjunto de casos de prueba.	9



Matrices de prueba

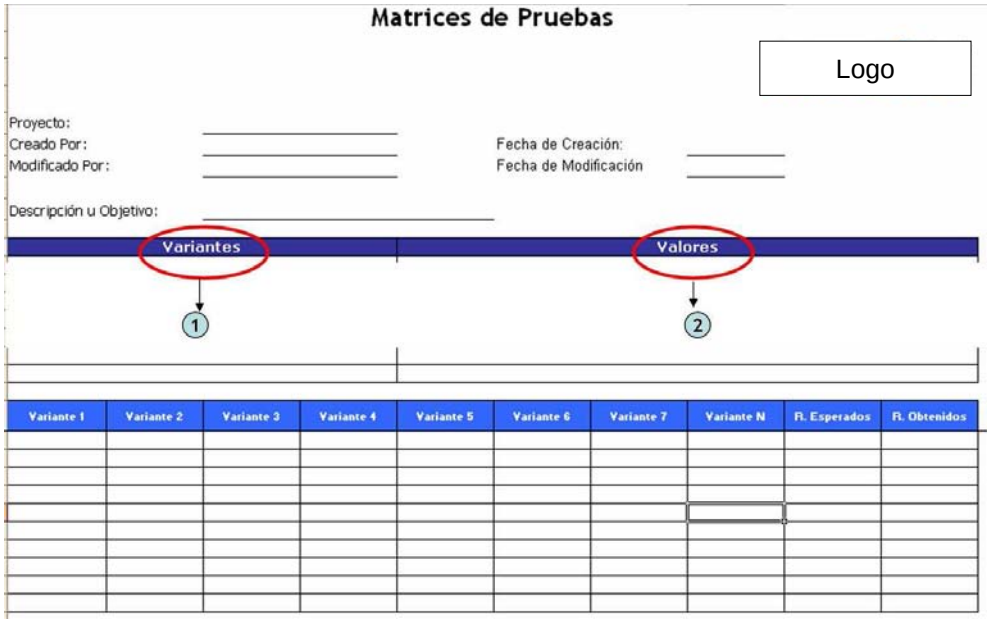
El formato para una matriz de pruebas sirve para realizar una combinatoria de todas las variantes de los casos de prueba con cada uno de sus valores, de tal forma de que no haya algún escenario sin validar por más remoto o poco probable que suceda.

Datos de Identificación:

- Proyecto
- Creado por
- Fecha de creación
- Modificado por
- Fecha de modificación
- Descripción u objetivo

Campos:

Nombre del campo	Descripción	Número
Variantes	Identificación de todas las variantes involucradas en el escenario a probar.	1
Valores	Identificación de todos los posibles valores que puede adquirir cada variante.	2



5.4.4. Post mortem

Reporte de post mortem

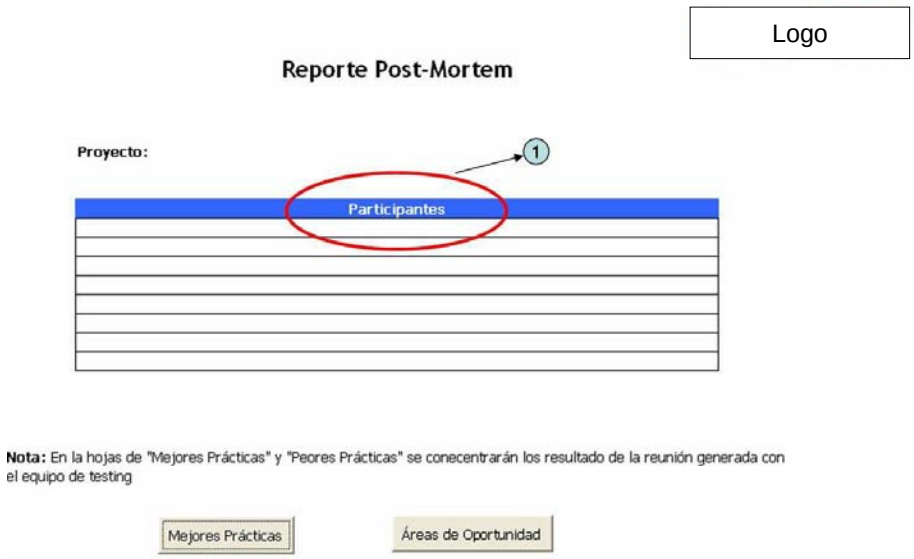
Formato diseñado para recopilar la información sobre las mejores prácticas y las áreas de oportunidad detectadas en un proyecto, para posteriormente poder publicarlo y robustecer el proceso de pruebas.

Datos de identificación:

- Proyecto

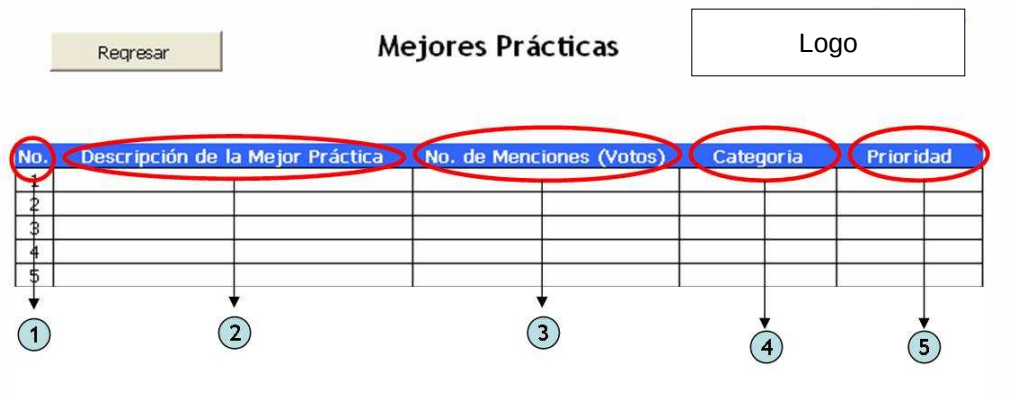
Campos (página principal):

Nombre del campo	Descripción	Número
Participantes	Nombre de los miembros del equipo y quienes están participando en la reunión post mortem	1



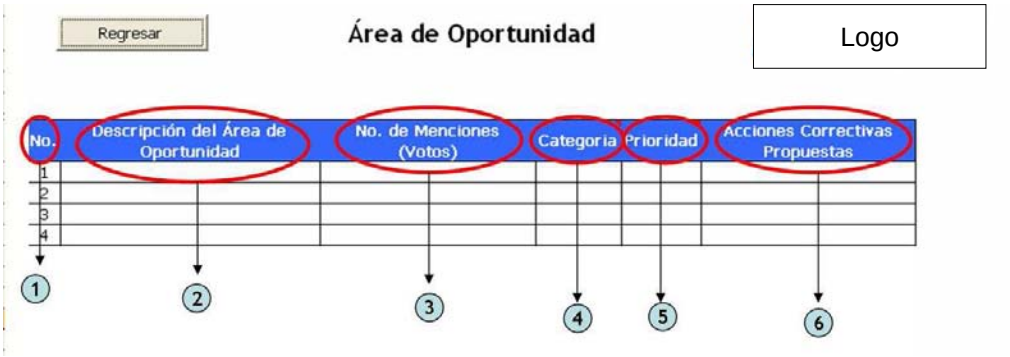
Campos (mejores prácticas):

Nombre del campo	Descripción	Número
No.	Número identificador de cada mejor practica por proyecto.	1
Descripción de la mejor práctica	Descripción detallada de la mejor practica mencionada en la reunión post mortem.	2
No. de menciones	Número total de votos que obtuvo la mejor practica en la reunión post mortem.	3
Categoría	División en la cuál se encuentra la mejor práctica (Ej. comunicación, liderazgo, administración)	4
Prioridad	Prioridad que se le asigna a la mejor práctica dependiendo de la importancia que ésta tenga.	5



Campos (áreas de oportunidad):

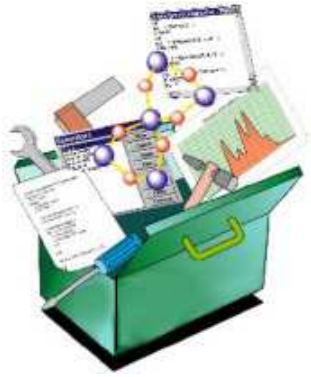
Nombre del campo	Descripción	Número
No.	Número identificador de cada mejor practica por proyecto.	1
Descripción del área de oportunidad	Descripción detallada de la mejor práctica mencionada en la reunión post mortem.	2
No. de menciones	Número total de votos que obtuvo la mejor práctica en la reunión post mortem.	3
Categoría	División en la cuál se encuentra la mejor práctica (Ej. comunicación, liderazgo, administración)	4
Prioridad	Prioridad que se le asigna a la mejor práctica dependiendo de la importancia que ésta tenga.	5
Acciones correctivas	Descripción de las acciones correctivas propuestas para convertir esta área de oportunidad en una mejor práctica	6



5.4.5. Entregables del proceso de pruebas

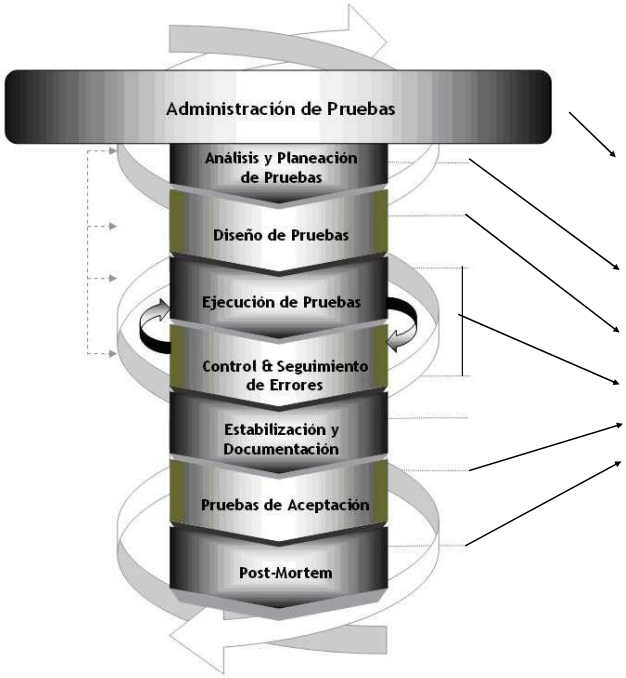
Los principales entregables que se deben considerar dentro del proceso de pruebas diseñado son:

- ✓ Plan de trabajo
- ✓ Especificación de pruebas
- ✓ Configuración ambiente de pruebas
- ✓ Matrices y casos de prueba
- ✓ Procedimientos y/o mapas de pruebas
- ✓ Acta de aceptación de pruebas
- ✓ Reporte de post mortem
- ✓ Bitácora de pruebas
- ✓ Control de riesgos
- ✓ Control de pruebas
- ✓ Reporte y control de versiones
- ✓ Reporte de errores y seguimiento
- ✓ Reporte semanal de actividades con avances



Nota: Los entregables a utilizar dentro del desarrollo de sistemas (arriba mencionados) se pueden delimitarán de acuerdo a las necesidades especificas de cada proyecto.

A continuación se muestra gráficamente la distribución de entregables dentro de las fases del proceso de pruebas:



Artefacto	Responsable
• Bitácora de Pruebas • Control de Riegos • Control de Pruebas •Reporte y Control de Versiones • Minutas	QA, BSA, BA QA, BSA, BA QA, BSA, BA, Usuario BSA, QA BA, BSA, Usuario, QA
•Especificación de Pruebas • Plan de Pruebas • Configuración Ambiente Pruebas	BSA, QA, BA, Usuario BSA, QA, BA, Usuario BSA, CA, QA
•Casos de Prueba • Matriz de Prueba	BA, Usuario, BSA, QA BA, Usuario, BSA, QA
• DTS	QA
• Acta de Pruebas de Aceptación	
• Reporte Post Mortem	QA, BA, BSA, Usuario

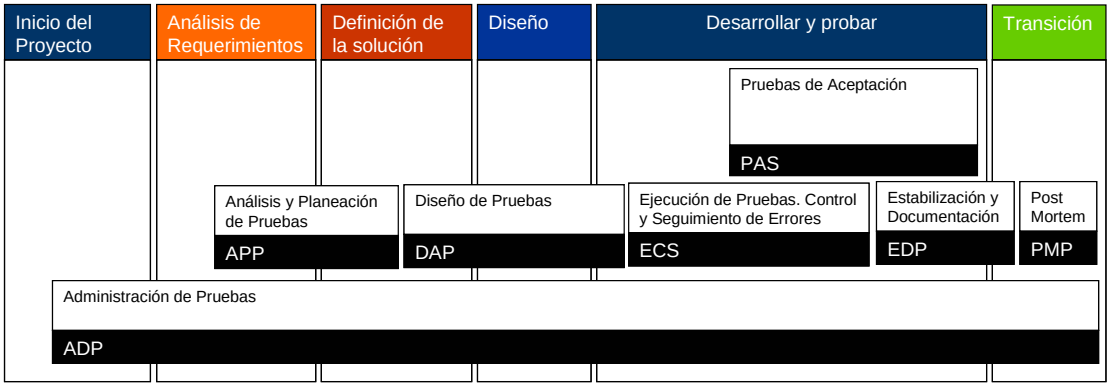
5.5. ESTRATEGIA DE IMPLEMENTACIÓN

- Implementar los nuevos procesos de verificación y validación durante la definición y construcción de los productos de software del proyecto.
- Planear y ejecutar auditorias de funcionalidad, configuración y de producto.
- Planear y ejecutar pruebas funcionales, de integración, de sistema, de volumen y de aceptación.
- Controlar todos los cambios y versiones de los productos de trabajo del proyecto.
- Se comienzan a utilizar elementos del nuevo proceso de pruebas en proyecto de desarrollo.
  - Especificación de pruebas
  - Casos de prueba
  - Matrices de prueba
- Aplicar la nueva metodología en los siguientes proyectos nuevos y mantenimientos a sistemas actuales.
- Impulsar la participación del área de aseguramiento de calidad desde el inicio de cualquier proyecto.

5.6. INTEGRACIÓN DE LAS FASES DE PRUEBAS DENTRO DE LAS METODOLOGÍAS DE DESARROLLO

Una vez integrado el nuevo proceso de pruebas en la metodología de desarrollo de sistemas, el diagrama de las fases quedo de la siguiente forma:

Estructura de módulos de pruebas para las fases de Desarrollo.



CAPÍTULO 6. PUESTA EN MARCHA

6.1. INTRODUCCIÓN

El presente capítulo muestra los programas y propuestas de capacitación que se deben impartir dentro de las organizaciones que deseen utilizar el proceso de pruebas propuesto. Concluyendo con una descripción de ventajas y beneficios que se obtienen con el uso del proceso de pruebas diseñado.

6.2. CAPACITACIÓN

¿Por qué capacitarnos?

"Si la capacitación es cara, la ignorancia lo es más". Anónimo.

La capacitación es una herramienta valiosa, que hoy por hoy se ha convertido en el instrumento más eficaz para realizar cambios y aumentar las competencias individuales, aspectos que benefician a las organizaciones y los podemos obtener con relativa facilidad, ya que algunas instituciones privadas los ofrecen. La capacitación nos aumenta la calificación de nuestra capacidad profesional y nos permite ser más productivos.

Debemos capacitarnos porque a través de la capacitación accedemos rápidamente a conocimientos y procedimientos que nuestra organización necesite para su desarrollo y que día a día nuestros clientes nos exigen. La capacitación también nos hace competitivos; mejora las habilidades para hacer un trabajo; incrementa nuestros valores humanos y sociales; acrecienta nuestro espíritu emprendedor y ganador; aumenta la autoestima y, finalmente, mejora nuestra calidad de vida. El conocimiento adquirido es algo que no pesa; todo lo contrario, nos sirve de medio efectivo para obtener ventajas y grandes beneficios en forma integral en todos los ámbitos en los que interaccionamos.

Por lo anterior, se tiene como objetivo de esta sesión, mostrar los programas de capacitación sobre fundamentos de pruebas y el nuevo proceso de pruebas así como presentar los temarios de dicho entrenamiento.

6.2.1. Temarios de los programas de capacitación

CURSO: "The next generation software testing workshop"

Justificación: Hoy en día los responsables de las pruebas de software se enfrentan, cada vez más, a la difícil tarea de proporcionar información sobre el grado de calidad de un software, dentro de ciclos de desarrollo cada vez más cortos. Las pruebas de software generalmente se entienden como la etapa de los proyectos de desarrollo, destinada a encontrar los errores del sistema; sin embargo, las pruebas de software son mucho más que eso y deben ser realizadas en diferentes instancias del proyecto

Descripción: Este seminario de 6 días está enfocado al conocimiento detallado del proceso de pruebas de software, cubre aquellos aspectos relevantes que un ingeniero de pruebas debe conocer para desarrollar mejor su trabajo. Así mismo, se enseñan los procesos, etapas, técnicas y tipos de pruebas que existen hoy en día, incluyendo recomendaciones al establecer y ejecutar pruebas de software.

Objetivos y beneficios: Al finalizar este taller los participantes serán capaces de:

- Presentar el proceso efectivo de pruebas de software y las metodologías relacionadas que mostrarán cómo hacer un proceso de pruebas adecuado y exitoso, en los proyectos de desarrollo de software.
- Identificar el ciclo de vida de las pruebas de software y su relación con el ciclo de vida de desarrollo de software.
- Identificar y describir las actividades, y los diferentes tipos de pruebas de software. así como saber aplicar las técnicas y tipos de pruebas necesarias de forma adecuada y oportuna de acuerdo a la etapa de pruebas en la que se encuentre el software.
- Conocer el proceso formal y adecuado para la realización de pruebas en un proyecto de software, lo que significa, una mejora del proceso actual de pruebas.
- Tener mayor conciencia sobre la necesidad imperante de tener un proceso de pruebas formal y definido, que permita registrar oportunamente los errores existentes garantizando su rápida y efectiva corrección.

Audiencia: Recomendado para aquellos que desean implantar y/o mejorar procesos de pruebas en su organización. Esto incluye, ingenieros de pruebas, gerentes o administradores de proyecto, y líderes de equipo responsables de interactuar con los ingenieros de prueba o que necesitan conocer más a fondo los pormenores del ¿Qué, cuándo y cómo? del mundo de las pruebas de software.

Módulo 1 – Fundamentos de pruebas

- ¿Por qué es importante probar?
  - Casos más famosos de errores
- Definición y características de pruebas
- Propósitos de las pruebas de software
- Ventajas y desventajas
- Estructura de un equipo de pruebas
  - Rol del ingeniero de pruebas de software *(¿Cuáles son sus funciones?)*
- Costo - Beneficio de las pruebas
- ¿Qué probar?
- ¿Cuándo probar?
- Etapas, técnicas y tipos de pruebas
- ¿Cómo probar?

Módulo 2 – Análisis y planeación de pruebas

- Analizar la documentación existente
- Determinar el alcance de pruebas
  - Especificación de pruebas
- Realizar pruebas de requerimientos
- Definición de escenarios de pruebas
- Generar el plan de trabajo de pruebas
- Estimación de pruebas
- Configurar el ambiente de pruebas

Módulo 3 – Diseño de pruebas

- Procedimiento de pruebas
- Mapas de pruebas
- Casos de pruebas
- Matrices de pruebas
- Scripts de pruebas
- Tips de pruebas manuales vs. automatizadas
- ¿Cómo documentar el diseño de pruebas?

Módulo 4 - Ejecución de pruebas y control de errores

- Métodos manuales
  - Riesgos de la NO automatización
- Automatización de pruebas
  - Riesgos de la automatización
- Herramientas
- Evaluación de pruebas
- ¿Qué es un bug?
- Elementos de control de errores
- Herramientas de control de errores

Módulo 5 – Automatización de pruebas

- Proceso de automatización de pruebas
  - ¿Qué es automatizar pruebas?
  - ATLM - The Automated Testing Lifecycle Methodology)
  - Beneficios y costos de la automatización
- Metodologías de automatización de pruebas
  - ¿Por qué automatizar?
  - ¿Qué automatizar?
  - ¿Cuándo tendrá éxito la automatización?
  - Decisión para automatizar las pruebas
- Herramientas apropiadas
  - Herramientas y tecnologías líderes en el mercado
  - Comparación y evaluación de herramientas
  - Elección de la mejor herramienta
- Ejecución automática de pruebas
  - ¿Qué es un script de pruebas?
  - Confiabilidad y validez de los scripts de pruebas
  - ¿Puedo ejecutar una serie de scripts de prueba?

Módulo 6 – Fases complementarias del proceso de pruebas

- Administración de pruebas
  - Organización de actividades
  - Riesgos
  - Métricas
  - Reporte de pruebas
  - Control de versiones
- Estabilización y documentación
  - Revisión final

  - Actualización
- Post mortem
- eXtend del proceso de pruebas
  - Probando bajo presión
  - Probando sistemas sucios

Módulo 7 – Conclusiones

- Mejores prácticas
- Recomendaciones
- Conclusiones
- “Las pruebas de software del mañana”

Nota: Cabe mencionar que dentro del taller se realizarán prácticas y demostraciones.

CURSO: “Seminario nuevo proceso de pruebas”

Justificación: Con el propósito de contar con mayor calidad en los desarrollos de la organización se llevó a cabo la implantación de un proceso formal de pruebas con el que garantizarán la liberación de nuevas aplicaciones prácticamente libres de errores. Por lo anterior, es necesario dar a conocer esta nueva metodología de pruebas, incluyendo procesos, artefactos, equipos de trabajo, etc.

Descripción: Este seminario de 3 horas esta enfocado al conocimiento detallado de los nuevos procesos de pruebas y su integración a la metodología de desarrollos de software.

Objetivos y beneficios: Al finalizar este seminario los participantes serán capaces de:

- Conocer el proceso efectivo de pruebas de software implantado y adecuado a las necesidades de la organización.
- Conocer la participación del equipo de aseguramiento de calidad en desarrollo de proyectos.
- Identificar las actividades de pruebas que serán ejecutadas en cada proyecto, así como conocer la aplicación de dichas actividades para las diferentes etapas de pruebas del proyecto.
- Destacar la importancia de llevar a cabo este proceso de pruebas para garantizar proyecto exitoso y sin defectos en producción, para lo cual se requiere de constante apoyo y cooperación de todas las personas involucradas en el proyecto.

Audiencia: Recomendado para aquellos involucrados en cualquier etapa del desarrollo de proyectos (área de TI).

Seminario nuevo proceso de pruebas

- Introducción a las pruebas
- Antecedentes (situación previa a la implantación del proceso de pruebas)
- Integración del proceso de pruebas a la metodología de desarrollo
- Explicación de las etapas del proceso de pruebas
- Procesos de prueba (nuevos proyectos y mantenimientos)

6.3. RESULTADOS DE LA CAPACITACIÓN

Este apartado tiene como objetivo dar a conocer los resultados esperados después de la capacitación propuesta que deben tomar los integrantes del equipo de pruebas.

Las ventajas competitivas que se obtienen son:

- Se proporciona una panorámica amplia de la prueba de software permitiendo a las organizaciones profundizar en metodologías y prácticas que al ser aplicadas en sus procesos de desarrollo, ayudan a la mejora de la calidad de productos de software.
- La organización se especializa en procesos de pruebas.
- Información proporcionada permite a las organizaciones que puede implementar un proceso de pruebas de manera ágil.
- Cada uno de los integrantes se involucran entre sí, obteniendo un mejor resultado entre todos.

6.4 VENTAJAS

El nuevo proceso de pruebas es completo y estable adoptado por el área de Tecnología de la Información para garantizar el correcto funcionamiento de las nuevas aplicaciones que se desarrollen, cambios y mejoras que cumplan con las necesidades y requerimientos de los usuarios, así como la disminución del riesgo de fallas una vez que la aplicación se encuentre en producción.

Nuestra meta:
"Encontrar errores y asegurarse de que sean corregidos"

Con este proceso se tienen las siguientes ventajas:

- Contar con un proceso de pruebas estable, que garantice el buen funcionamiento del software.
- El staff de pruebas podrá aplicar los conocimientos adquiridos en la implementación dentro de un proyecto real y así reforzar todo lo aprendido.
- Garantizar que los requerimientos del cliente sean cumplidos (necesidades de negocio y operación).

- Optimizar el proceso de corrección de errores (durante todo el desarrollo del proyecto en sus diferentes etapas).
- Encontrar los errores antes de que los encuentre el usuario (esto evita que el usuario desconfíe del sistema y no tenga la aceptación esperada).
- Tener documentación de todas las pruebas realizadas, así como estadísticas de tiempos de ejecución de pruebas y corrección de errores.
- Todo el proyecto estará soportado por un "staff de pruebas" y no solo por un Ingeniero de pruebas.
- Contar con una diversidad de pruebas aplicadas a los sistemas con la finalidad de incrementar su calidad.

6.5. BENEFICIOS

El proceso de pruebas implementado cubre el alcance total del proyecto, desde la gestión de la calidad del proyecto, la gestión de los riesgos asociados, la definición de la estrategia de pruebas, el plan de pruebas y todas aquellas actividades y tareas directamente relacionadas con garantizar el aseguramiento de calidad del desarrollo de los sistemas de información.

Algunos de los beneficios que se obtienen al implantar el nuevo proceso de pruebas dentro de la metodología de desarrollo de sistemas de las áreas de Tecnología de la Información son:

Para los clientes:

- *Costo de oportunidad controlado*. Dependiendo de la importancia de la aplicación solicitada, no contar con la misma en el momento previsto, puede ocasionar pérdidas considerables, tanto económicas como de imagen.
- *Eficiencia en la operación del día a día*. Contar con una aplicación desarrollada bajo estándares de calidad asegurados, garantiza la estabilidad del software, evitando interrupciones en las actividades del negocio por defectos del sistema.
- *Aumento en la productividad*. Una aplicación bien diseñada y desarrollada, facilita las actividades de trabajo diarias, aumentando la productividad en tareas administrativas, productivas y de control entre otras.
- *Reducción en los costos operativos*. La implementación de software de calidad, evita costos asociados a eventos tales como caídas del sistema, demoras en la atención a clientes, pérdidas de información vital del negocio.
- Se prevé mayor cobertura en las pruebas para asegurar la correcta funcionalidad de los sistemas antes de salir a producción.

Para las áreas de Tecnología de Información:

- *Reducción de costos de desarrollo*. Principalmente costos asociados a la no calidad, que se traducen en muchas horas dedicadas a corregir errores de aplicaciones que ya está en producción, necesidad de más recursos humanos para poder cumplir con los plazos establecidos para la finalización de los proyectos y, quizás el más grave, pérdidas económicas a nivel negocio por errores funcionales y conceptuales en las aplicaciones.

- *Clientes internos satisfechos*. Porque entregar software en tiempo y alineado con las expectativas del cliente o usuario, genera una imagen de profesionalismo del área de TI y trasmite confianza al resto de la organización.

- *Mayor disponibilidad de recursos humanos*. Porque al eliminar los tiempos invertidos en retrabajo, por cuestiones asociadas a la no calidad, baja considerablemente el número de horas invertidas en cada proyecto, liberando anticipadamente los recursos asignados a un determinado proyecto y dejándolos disponibles para comenzar los próximos.

- *Mejor organización e integración de los equipos de trabajo*. Desarrollar software en base a un proceso estandarizado y repetitivo, permite controlar eficientemente varios equipos de trabajo.

- Integración de mejores prácticas, patrones y estándares de desarrollo de software de calidad mundial.

- Por el conocimiento de diferentes metodologías se tiene facilidad de entendimiento y adopción de la metodología a aplicar.

- Identificación y seguimiento de hallazgos.

- Se facilita la definición y el cumplimiento de los objetivos del área de calidad.

- Reducción en los costos de ejecución de pruebas de regresión.

- Mayor dedicación a la mejora continúa de los procesos garantizando que estas mejoras se apeguen al Marco de referencia que eligió la organización.

- Contar con material de apoyo para garantizar el entendimiento de los procesos y procedimientos organizacionales.

- Administración de los proyectos de pruebas.

- La generación de propiedad intelectual y prácticas de negocio.

CONCLUSIONES

¿Es bueno su proceso de pruebas?

Esta pregunta aparentemente tan fácil puede resultar difícil de responder en la práctica. El proceso de realización de pruebas se percibe con frecuencia como un proceso problemático e incontrolable. Realizar pruebas conlleva mucho más tiempo, cuesta mucho más de lo planeado, y ofrece insuficiente información sobre la calidad del proceso de pruebas y, por lo tanto, de la calidad del sistema de información que está siendo probado, así como de los riesgos para el proceso de negocio en sí mismo.

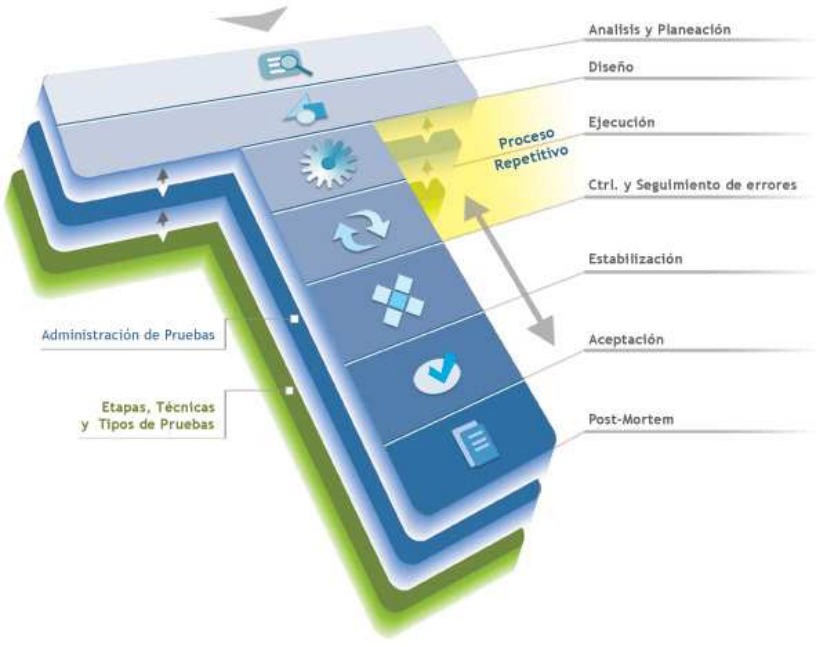
Muchas organizaciones se dan cuenta de que mejorar el proceso de pruebas puede resolver estos problemas. Sin embargo, en la práctica, resulta difícil definir qué pasos hay realizar para mejorar y controlar dicho proceso, y en qué orden deben ser realizados.

Los desarrollos se producen en la actualidad a una gran velocidad. Los productos desarrollados están en continuo crecimiento y los clientes cada vez exigen mayor calidad. Aún cuando el proceso actual de pruebas sea satisfactorio, este proceso necesitará mejorar en el futuro.

Con el nuevo proceso de pruebas implementado se puede concluir que al emplear de forma sistemática y disciplinada modelos, métodos y herramientas de Ingeniería de software para el aseguramiento de la calidad favorece no sólo la comprensión y análisis, sino que potencializa la mejora de la calidad producida. La metodología de desarrollo de sistemas combinada con el nuevo proceso de pruebas, proporciona un enfoque práctico, sistémico y cuantitativo para la evaluación de la calidad funcional de los sistemas de información.

Por otra parte, el empleo de herramientas que brinden soporte a la metodología permite a los responsables de las pruebas agilizar los procesos y minimizar las situaciones de riesgo e imprecisiones.

Algo muy interesante del nuevo proceso de pruebas es que al tener todo el testware documentado y con la creación de un correcto diseño de pruebas (procedimientos, mapas, casos, matrices y scripts de pruebas) provee todo lo necesario para su reutilización.



APÉNDICES

I. PREGUNTAS MÁS FRECUENTES

1. **¿Cuál es la meta de un ingeniero de pruebas?**
   Su meta es encontrar errores lo más pronto posible, y asegurarse de que sean corregidos.

2. **¿Un buen ingeniero de pruebas debe buscar siempre la perfección del producto de software?**
   No, en realidad, un buen ingeniero de pruebas sabe cuándo es inalcanzable la perfección, y sabe definir "cuándo ha sido suficiente" en sus alcances dentro del proceso de pruebas.

3. **¿Es importante la terminología que utilice una organización para el manejo de problemas en su software?**
   No, las organizaciones pueden nombrar al manejo de los problemas que se presenten en su software de cualquier forma; sin embargo, es importante tomar en cuenta que su terminología reflejará la personalidad de su equipo y la manera en la que encuentran, reportan y arreglan dichos problemas.

4. **¿Cuál es el problema de probar que un programa trabaje sólo como se espera?**
   El problema de probar sólo lo que se espera del programa, es que nada más se estaría abarcando la mitad de las pruebas, ya que, por lo general, el usuario no sigue las reglas del sistema y es necesario que los ingenieros de pruebas prueben que el sistema siga funcionando en caso de que esto suceda. Además, si un equipo de pruebas no tiene la filosofía de buscar la forma de tronar el sistema, se les pueden escapar una gran cantidad de errores.

5. **¿Qué costo genera detectar un error una vez que se ha liberado el sistema, comparado con detectarlo en las etapas tempranas del proyecto?**
   Aproximadamente de 10 a 100 veces o más de lo que pudo costar arreglarlo durante el proyecto.

6. **¿Qué es administración de pruebas?**
   Es la disciplina de aplicar los principios de ingeniería de desarrollo, implementación y seguimiento de un proyecto.

7. **¿Cuáles son los principios de ingeniería de procesos?**
   Los proyectos y riesgos de administración, diseño de alto y bajo nivel, y colección de métricas.

8. **¿Quién se encarga de realizar las pruebas unitarias?**
   El desarrollador o developer.

9. **¿Qué es la prueba de requerimientos?**
   Inspección en la cual se busca evaluar la consistencia, completitud y factibilidad de los requerimientos, tanto individualmente como juntos.

10. **¿Qué son las pruebas de integración?**
    Son pruebas que se realizan al unir los diferentes componentes del sistema y revisar su comportamiento así como su buen funcionamiento.

11. **¿Cuáles son las pruebas que se realizan en todo el sistema por el cliente (ó supervisión de este), de preferencia desarrolladas en un ambiente totalmente operacional?**
    Las pruebas de aceptación.

12. **¿Qué son las pruebas dinámicas?**
    Son pruebas realizadas con el programa en ejecución tal y como lo utilizaría el usuario.

13. **¿Qué pruebas se realizan directamente sobre la interfaz de usuario manipulando los datos de entrada y revisando los resultados?**
    Pruebas de caja negra.

14. **¿Cuáles son las razones más importantes por las que la especificación del producto es generalmente la fuente que genera más errores en un producto de software?**
    Si no se determinan todas las especificaciones, simplemente no se puede hacer el producto con las características requeridas.
    Otra razón es que, muchas veces sí están escritas las especificaciones, pero no se comunican a todo el equipo de desarrollo o de pruebas los cambios que se lleven a cabo, por lo que, es como si no se hubieran hecho correctamente las especificaciones.

15. **Ejemplos de requerimientos no funcionales serían:**
    Carga, estrés, usabilidad, volumen, documentación, desempeño y seguridad.

16. **¿Es posible tener una gran calidad y baja confiabilidad en un producto?**
    Si es posible, pero depende mucho de las expectativas de calidad que tiene el cliente. Por dar un ejemplo, muchas personas compran autos deportivos que consideran de alta calidad por su potencia, velocidad, estilo, ajuste, etc. Sin embargo, por lo general, estos mismos autos son no confiables, descomponiéndose o estropeándose y costando muy cara su compostura, pero sus dueños no consideraron esa poca confiabilidad como parte de la calidad.

17. **¿Por qué resulta imposible probar un producto de software completamente?**
    Aún tomando en cuenta al producto de software más pequeño y simple, existen muchas entradas y salidas, y muchas combinaciones que se pueden generar para una prueba absoluta.

18. **¿Es posible elaborar pruebas dinámicas de caja negra sin la especificación del producto o algún documento de requerimientos?**
    Sí es posible. La técnica se llama: “pruebas exploratorias”, y esencialmente se basa en considerar que el software que se está probando sí cumple con sus especificaciones. No es un proceso ideal, pero puede resultar en momento dado. El mayor riesgo es que no se puede saber si está faltando por cubrir alguna característica del producto.

19. **¿Cuáles son los tres puntos esenciales que debe contener todo diagrama de transición de estados?**
    - Especificar cada uno de los estados en los que el software se puede encontrar.
    - La entrada o condiciones que lo lleva cambiar de un estado al siguiente.
    - Condiciones, variables o salidas que se deben generar cuando un estado ha terminado o se ha salido de éste.

20. **¿Existen ventajas al desempeñar pruebas estáticas de caja blanca?**
    Si, ya que las pruebas estáticas de caja blanca permiten encontrar errores tempranamente en el ciclo de desarrollo. Los ingenieros de pruebas pueden obtener información de cómo trabaja el software, cuáles debilidades y áreas de riesgo existen, además de que pueden aprovechar para construir una mejor relación de trabajo con los desarrolladores.

21. **¿Qué diferencia existe entre un documento estándar y un documento pauta?**
    Un documento estándar define específicamente lo que se tiene y no se tiene que hacer, es decir, determina las reglas que se tienen que seguir. Un documento pauta son sugerencias de mejores prácticas, recomendaciones sobre el mejor camino a seguir para realizar algo.

22. **¿Por qué el conocer cómo trabaja el software influye en decidir cómo y qué se debe probar?**
    Cuando se toma en cuenta sólo una perspectiva de caja negra con respecto al software, no se podrá saber si los casos de prueba están cubriendo adecuadamente todas las áreas del software, en incluso tal vez se estén llevando a cabo casos de prueba redundantes. Una prueba de caja negra puede diseñar una suite de pruebas bastante eficiente sobre un producto de software, pero no se puede saber que tan buena es esa suite sin algún conocimiento de caja blanca ¿cómo trabaja el software internamente?

23. **¿Cuál es la diferencia entre pruebas dinámicas de caja blanca y debugging?**
    Ambos procesos son muy similares, los dos se encargan de ver exactamente cuándo y por qué ocurren los errores, pero la meta de las pruebas dinámicas de caja blanca es meramente encontrar errores, y la meta del debugging es arreglarlos.

24. **¿Cuál es el problema que se puede presentar en el uso de pruebas de caja blanca, ya sean estáticas o dinámicas?**
    El tester puede ver el código y llegar a concluir: “Veo el código, y no necesita caso de prueba, está elaborado correctamente”. En realidad, tal vez se pueda deslumbrar con el código y gracias a eso omitir casos de prueba verdaderamente importantes. Es necesario ser muy cuidadoso.

25. **¿A qué se refiere el hecho de configurar el ambiente de pruebas?**
    A llevar a cabo la configuración del hardware, software y otros requerimientos necesarios para las pruebas.

26. **¿Cómo se puede establecer si un error encontrado corresponde a un problema general o a un problema específico de configuración?**
    Una forma para saberlo es corriendo de nuevo los pasos para recrear el error en diferentes configuraciones. Si el problema no vuelve a ocurrir en éstas, entonces seguramente se tratará de un error de configuración. Si ocurre de nuevo en todas las configuraciones, entonces se trata de un problema general. Es muy importante probar en distintas configuraciones, por ejemplo, tal vez un error se puede generar cuando se prueba en impresoras láser, pero no en impresoras de inyección de tinta.

27. **¿Es posible liberar un producto de software con errores de configuración?**
    Si se puede liberar, debido a que no es posible arreglar absolutamente todos los errores, como en todo tipo de prueba. El equipo de pruebas deberá decidir qué se puede arreglar y qué no. Si se deja un error, que sólo ocurre con una extraña pieza de hardware, es fácil decidir que no hay problema en la liberación del producto; sin embargo, no todas las decisiones de liberación son así de sencillas.

28. **¿Cuáles son algunos tipos de formatos de datos que podrían variar dentro de un programa en el cambio de una localidad a otra?**
    Medidas como libras, pulgadas, galones. Tiempo en formato 24 horas o 12 horas. Moneda como euros, dólares, pesos.

29. **¿Todo software tiene interfaz de usuario y a ésta deben aplicarse pruebas de usabilidad?**
    Si, eventualmente todo software, aún el más embebido y profundo, es expuesto de alguna manera al usuario. Esto ocasiona que sea probado si la interfaz de usuario es tan simple como un apagador de luz o tan compleja como un transformador de luz, y esa apreciación la decidirá finalmente el usuario.

30. **¿Si no existe una interfaz de usuario definitiva, cómo puede ser probada?**
    Los ingenieros de pruebas deben concentrarse en tomar siete importantes criterios: que siga los documentos estándares o documentos pauta, que sea intuitivo, consistente, flexible, confortable, correcto y usable.

31. **¿Qué son las pruebas de caja gris?**
    Las pruebas de caja gris son aquellas que se efectúan cuando se da una ojeada al código más importante y se utiliza dicha información como apoyo para las pruebas. Es diferente de las pruebas de caja blanca porque se examina un código simple, no complejo como por ejemplo uno en C++. Además, no se hace el mismo nivel de detalle de revisión que se haría en pruebas de caja blanca.

32. **¿Qué beneficios ofrece el uso de herramientas de pruebas de software y automatización?**
    Algunos de sus beneficios, es la optimización de velocidad que se obtendría para correr los casos de prueba. Además, permite obtener resultados más precisos, seguros y rigurosos. Los ingenieros de pruebas pueden hacer el proceso más eficiente si se dedica tiempo a la planeación y desarrollo de pruebas.

33. **¿Cuáles serían técnicas de pruebas de forma manual?**
    Pruebas de escritorio, inspecciones, revisiones formales, entre otros.

34. **Ejemplos de herramientas de automatización.**
    Test Case / Script Generators, Network Monitors, Coverage Analysis, Defect Tracking, Herramientas de Compuware, etc.

35. **Algunos ejemplos de herramientas que se utilizan para el uso de máquinas virtuales son:**
    VMware, Connectix, entre otras.

36. **¿Cuál es la diferencia entre herramienta de pruebas y automatización?**
    Una herramienta de prueba ayudará al tester a realizar sus pruebas, haciendo más fácil la ejecución de las pruebas que se harían manualmente. Automatización es también una herramienta, pero la función de ésta es correr sin la intervención de un tester, es decir, dejarla corriendo y que haga el trabajo sola.

37. **Precauciones que se deben considerar para utilizar herramientas de pruebas de software y automatización, serían:**
    - Debido a que el software puede cambiar durante su desarrollo, las herramientas de pruebas necesitarían cambiar también.
    - Se puede llegar a perder mucho tiempo diseñando herramientas y automatizando.

    Es muy sencillo llegar a confiar demasiado en la automatización, situación que tampoco puede ser buena, debido a que no hay que perder de vista que no hay algo que pueda sustituir el realizar las pruebas uno mismo.

38. **¿Cuál es el porcentaje máximo que se recomienda automatizar en un proyecto?**
    No más del 50%.

39. **¿En qué consiste la paradoja del pesticida?**
    La paradoja del pesticida ocurre si las personas continúan probando software con las mismas pruebas, o la misma gente. Eventualmente, el software crea inmunidad a las pruebas porque no se encuentran nuevos errores. Si surge un cambio en las pruebas o se cuenta con personas nuevas para elaborar pruebas es muy probable encontrar más errores, mismos que han estado ahí todo el tiempo pero que no eran visibles para los primeros testers.

40. ¿Por qué es conveniente contratar outsourcing para pruebas en una pequeña empresa?
Porque el costo e inversión que se necesita para montar un laboratorio de pruebas es alto, por lo que una pequeña empresa probablemente no podría solventarlo.

41. ¿Cuál es el propósito de un plan de pruebas?
Basándonos en la definición de ANSI/IEEE, el propósito de generar un plan de pruebas es definir el alcance, acercamiento, recursos y tiempo de la actividad de pruebas; y para identificar los puntos a probar, las características a ser probadas, y las tareas de prueba a desempeñar, el personal responsable para cada actividad, y los riesgos asociados con el plan. En pocas palabras, establecer y comunicar el plan a seguir para llevar a cabo las pruebas del software a todo el resto del equipo que participe en el proyecto, y ver si están de acuerdo con ello.

42. ¿Cuáles son los objetivos de la especificación de pruebas?
Identificar la información existente del proyecto y los componentes de software que deberán ser probados; así como enlistar los requerimientos de alto nivel que serán probados, y recomendar y describir las estrategias de pruebas a ser utilizadas.

43. ¿Qué es un caso de prueba?
Un caso de prueba es un documento que describe la entrada de datos, una acción o un evento y la respuesta que se espera, y así poder determinar si la aplicación está funcionando correctamente. Un caso de prueba debe tener características tales como nombre del caso de prueba, objetivo, condiciones para prueba, datos de entrada, los pasos a seguir, y los resultados esperados.

44. ¿Qué beneficios tiene desarrollar casos de prueba?
El desarrollo de los casos de prueba puede ayudar a detectar problemas en los requerimientos o en el diseño de la aplicación, ya que requiere que se documente cada caso de todo el sistema. Por esta razón es recomendable que los casos de prueba se elaboren cuando se comienza la fase de desarrollo.

45. ¿Qué es la especificación de casos de prueba?
Este documento define los valores de entrada actuales para pruebas y lo que se espera en las salidas. Además, lista cualquier tipo de ambiente necesario o procedimiento que se requiera y cualquier interdependencia entre los casos de prueba.

46. ¿Qué artefactos, además de un documento tradicional, se pueden utilizar para mostrar los casos de prueba?
Tablas, matrices, listas, diagramas, gráficas, etc. Cualquiera será de mucha utilidad para aumentar la efectividad de presentación y manejo de casos de prueba, ya sea para el tester que los creó, o para el resto del equipo de pruebas.

47. ¿Qué son los scripts de prueba?
Son Instrucciones entendibles por la computadora que automatizan la ejecución de un método de prueba.

48. ¿Para qué se utiliza una matriz de pruebas?
Para describir el flujo a seguir del negocio, especificando al final de ésta cuáles son los resultados esperados del proceso, e indicando si el resultado de la prueba fue exitoso o no.

49. ¿Cuál es el propósito de la especificación del procedimiento de prueba?
El propósito de esta especificación es identificar todos los pasos requeridos para desempeñar los casos de prueba, incluyendo el cómo de la instalación, inicio, correr y terminar una prueba. También debe explicar qué hacer en caso de que una prueba no funcione como se planeó.

50. ¿Cuál fue el primer error encontrado en la historia?
En 1945. Una palomilla.

51. ¿Cuáles serían algunas razones por las que un error NO se corrija?
Que no haya suficiente tiempo, que no sea en realidad un error, que sea muy riesgoso arreglarlo, que el error no tenga mucha importancia para el sistema y que el error no haya sido reportado apropiadamente.

52. ¿Cuáles son los tres estados básicos que debe tener el ciclo de vida de un error, y estados adicionales más comunes?
Los estados básicos son: abierto o nuevo, resuelto y cerrado. Estados adicionales utilizados con más frecuencia son revisado, validado, rechazado o no aplica.

53. ¿Qué información debe llevar el reporte de errores?
La fecha, versión, prioridad, tipo, pasos para obtenerlo, localización, estatus, quién lo encontró, quién lo generó y quién lo solucionó.

54. ¿El equipo de pruebas es el encargado de la calidad del producto?
Definitivamente no. El equipo de pruebas busca errores, pero no los crearon dentro del producto y no pueden garantizar que las pruebas realizadas dejen al producto sin un solo error.

55. ¿Cuál es la diferencia entre validación y verificación?
"Verificar" generalmente se refiere a revisiones y juntas para evaluar documentos, planes, códigos, requerimientos y especificaciones. Esto se puede hacer con listas, inspección de código, etc. "Validar" son las pruebas en si y toma lugar después de que la verificación está terminada.

II. AUTOMATIZACIÓN DE PRUEBAS

Características:

- **Más proyectos / menos tiempo (minimizar costos):** Uno de los dolores de cualquier equipo de pruebas es el poco tiempo del que disponen para probar, por lo tanto automatizando podemos alcanzar mayor cobertura de las aplicaciones a probar en menos tiempo, de ahí la reducción de costos al no liberar aplicaciones con errores ni tampoco perder el "*time to market*".

- **Proyectos complejos:** Existen proyectos y/o requerimientos que por su naturaleza será indispensable utilizar herramientas debido a que no existe forma o sería muy complicada una verificación manual (Por ejemplo: pruebas de performance).

- **Requerimientos cambiantes:**
  - Una de las mejores prácticas al automatizar es contemplar en nuestras rutinas, el "flujo básico" para que en caso de que existan cambios dentro de la aplicación se pueda probar rápidamente que el sistema no fue afectado.
  - Otro punto importante es que con la automatización es factible adecuar funcionalidad de manera que si algo cambia puede ser más fácil generar una rutina para verificar su correcto funcionamiento.

- **Más pruebas (más robustas y diferentes):** Como se ha mencionado en características anteriores el uso de herramientas nos va a permitir realizar pruebas más complejas y que manualmente serían imposibles y/o muy difíciles de aplicar.

- **Reducción de riesgos.** Al tener automatizado parte de nuestro testware (y sobre todo bien enfocado) podemos ejecutar más ciclos de pruebas en los requerimientos de negocio con mayor riesgo ó más importantes, reduciendo así la probabilidad de falla.

- **Reusabilidad:** Con la experiencia se pueden generar rutinas que pueden ser utilizadas en diferentes proyectos, reduciendo así el tiempo de diseño y ejecución de pruebas.

- **Mayor calidad:** Si probamos más veces y a diferentes niveles estaremos asegurando mayor calidad en los productos.

Además de las características antes mencionadas es necesario tener en cuenta lo siguiente:

Riesgos de no automatizar.

- **Disminuye la cobertura de pruebas.** Al hacer todas las pruebas manuales se puede correr el riesgo de no poder cubrir toda la funcionalidad del sistema o no hacer las suficientes pruebas debido a la complejidad o a la premura del tiempo.

- **Ciclos de pruebas muy largos.** Se puede consumir mucho tiempo el ejecutar todas las pruebas una y otra vez por cada ciclo, además se complica si algunos módulos de la aplicación son muy complejos o tienen muchas variantes.

- **Aumenta el riesgo de fallas.** Muchas veces al no contar con el tiempo o con una forma de validar aspectos del software que no se pueden hacer manual (por ejemplo, performance) se puede correr el riesgo de que una vez en ambiente productivo comience a fallar por no haber podido validar estas condiciones.

Consideraciones al automatizar.

- **No es recomendable automatizar más del 50% de un proyecto.** La creación de scripts que pretendan cubrir la mayor parte de la funcionalidad de un sistema puede dar como resultado la inversión de mucho tiempo durante la ejecución y mantenimiento de éstos, cuando hacer las pruebas de forma manual puede resultar más confiable y/o práctica en algunos casos.

- **Eliminar pruebas manuales.** Siempre se va a requerir personal capacitado como son los ingenieros de pruebas para aportar su conocimiento, experiencia e intuición al aseguramiento de calidad, elementos que una herramienta no puede proporcionar.

- **Establecer expectativas irreales.** La automatización no va a sustituir el trabajo de los profesionales de pruebas, sino que es un complemento que los ayudará a asegurar la calidad de un sistema. Las herramientas no serían capaces de reportar todos los errores existentes o de cubrir todos los casos de uso o funcionalidad de la aplicación.

Herramientas

Dentro de la automatización de pruebas podemos encontrar diferentes tipos de herramientas que nos ayudarán a verificar diferentes aspectos de la aplicación que estamos probando; por mencionar algunas:

- Memory Leak Detectors.
- Test Data Generators.
- Test Case / Script Generators.
- Network Monitors.
- Defect Tracking.
- Performance (Load) Tools.

Los 10 desafíos de la automatización

Es recomendable contemplar los siguientes puntos al utilizar herramientas y automatizar pruebas:

1. Adquirir la herramienta correcta.
2. Tener un equipo adecuado de pruebas en la organización.
3. Correcta integración entre proceso y herramientas.
4. Cobertura correcta de tipos de pruebas.
5. Capacitación formal en la herramienta.
6. Correcta implantación de la herramienta.
7. Procesos básicos de pruebas y entender lo que se va a probar.
8. Correcto mantenimiento y configuración de scripts (reusabilidad).
9. Compatibilidad e interoperabilidad correcta de la herramienta.
10. Disponibilidad y accesibilidad de herramientas.

Relación entre las pruebas manuales y automatizadas

- Los dos métodos se complementan.
- Cada una tiene sus ventajas (fortalezas) y debilidades.
- Los ingenieros de pruebas de software que realizan pruebas manuales no corren riesgo al llegar la automatización.
- Permitir que cada una haga lo suyo y para lo que es mejor.

II.1. HERRAMIENTAS

Actualmente podemos delimitar los siguientes tipos de herramientas de pruebas:

- Test Management Tools
- Test Data Generators
- Network Monitors
- Defect Tracking
- Herramientas de automatización de pruebas (Robots)
  - *Performance (Load) Tools*
  - *Record /Playback Class Tools*

Algunas recomendaciones al seleccionar una herramienta:

- Identifique las áreas que pueden ser automatizadas y planee su esfuerzo de la automatización. Generar una lista de los requerimientos funcionales que se buscan en la herramienta.
- Planear una sesión de "lluvia de ideas" con los integrantes del equipo para priorizar los requerimientos.
- Desarrollar un cuestionario con las conclusiones de los requerimientos contra las características de las herramientas que existen en el mercado.
- Descargar y/o probar versiones de prueba de por lo menos tres herramientas que cumplan con los requerimientos que fueron delimitados con anterioridad.
- Revisar comentarios, información y recursos de especialistas que hayan utilizado las herramientas.

Una vez con la herramienta de automatización:

- Establezca la arquitectura de la automatización.
- Busque generar "scripts" reutilizables.
- Obtener métricas de la automatización (por ejemplo: tiempo de desarrollo, tiempo ejecución de los scripts, cobertura de pruebas, etc.)

Es importante tener en cuenta que la automatización de pruebas es "desarrollo" y por lo tanto es necesario contemplar, diseño, estándares de codificación, control de versiones, etc.

GLOSARIO

Glosario con términos y definiciones de la calidad, pruebas de software, ingeniería de requerimientos, métodos formales, etc.

A

- **Adaptabilidad** (*Adaptability*). Característica de portabilidad, que indica las características del software que influyen en las posibilidades de adaptación a diferentes entornos especificados, sin realizar otras acciones que las indicadas para este propósito.

- **Administración de calidad del proyecto.** Es la actividad derivada de la administración de proyectos, donde se realizan los procesos necesarios para llevar a cabo el proceso de manera satisfactoria, es decir que cumpla con los objetivos para los que fue creado. Consiste llevar a cabo un control de calidad eficiente y efectivo.

- **Administración de costos del proyecto.** Es la actividad derivada de la administración de proyectos, donde se realizan los proceso necesarios para llevar a cabo el proceso dentro del presupuesto contemplado para el. Esta consiste de planeación de recursos, estimación de costos, presupuestación de costos, y control de costos.

- **Administración de la configuración.** Disciplina que aplica dirección y vigilancia tanto tecnológica como administrativa para la identificación y documentación de las características físicas y funcionales de un elemento de configuración. Lleva el control de los cambios en dichas características, así como el registro y reporte de cambios en el proceso e implementación. También verifica el apego de la configuración a los requerimientos especificados.

- **Administración de proyectos.** Es el proceso de planear, organizar, dirigir y controlar el uso de recursos para lograr objetivos, que se plantean desde un principio por los involucrados en el proyecto.

- **Administración total de calidad (TQM).** Una aproximación común para implementar un programa de mejoramiento de la calidad dentro de una organización.

- **Ambiente operativo:** Es el estado de una computadora, determinado por los programas que se están ejecutando, hardware básico y características del software.

- **Análisis.** Es el primer paso dentro del desarrollo de sistemas, en el cual el analista se reúne con el cliente y/o usuario (un representante institucional, departamental o cliente particular), e identifican las metas globales, se analizan las perspectivas del cliente, sus necesidades y requerimientos sobre la planificación temporal y presupuestal, líneas de mercadeo y otros puntos que ayudan a la identificación y desarrollo del proyecto.

- **Análisis dinámico.** El proceso de evaluar un sistema o componente basándose en su comportamiento durante la ejecución.

- **Análisis estático.** El proceso de evaluar un sistema o componente basándose en su forma, estructura, contenido o documentación.

- **Análisis de ruta.** Análisis de un programa para identificar todos los posibles caminos dentro de éste. Así se detectan rutas incompletas o porciones de un programa que no están en algún camino de ejecución.

- **Analizador de tiempo.** Herramienta de software que estima o mide el tiempo de ejecución de un programa o porción del mismo, ya sea mediante la suma del tiempo de ejecución de cada instrucción en rutas previamente especificadas o con el sondeo en puntos específicos del programa.

- **Anomalía.** Cualquier observación en la documentación u operación del software que no está dentro de las especificaciones previamente establecidas.

- **Arnés de pruebas** (*test harness*). Un programa o script que permite la ejecución y la secuenciación de los casos de pruebas.

- **Arquitectura del conjunto de pruebas.** Las relaciones establecidas que se dan entre los conjuntos de casos de prueba que reflejan directamente una descomposición jerárquica de los objetivos de las pruebas.

- **Artefacto de software** (*software artefact*). Cualquier entregable que resulte del proceso de desarrollo de software, por ejemplo documentos de requerimientos, especificaciones, diseños, software, etc.

- **Aseguramiento de la calidad** (Quality Assurance). Conjunto de todas las acciones necesarias para asegurar que un elemento o producto se apegue a requerimientos técnicos ya establecidos.

- **Aserción.** Expresión lógica que especifica un estado que debe existir dentro de un programa, o un conjunto de condiciones con variaciones del programa que deben satisfacer un punto particular durante la ejecución del programa.

- **Aserción de entrada.** Expresión lógica que especifica una o más condiciones que las entradas de un programa deben satisfacer para considerarse válidas.

- **Aserción de salida.** Expresión lógica que especifica las condiciones que la salida de un programa debe cumplir para considerarse correcta.

- **Auditar.** Una revisión independiente de un producto trabajado, o del conjunto de productos trabajados para determinar la conformidad de especificaciones, estándares, acuerdos contractuales, o cualquier otro criterio.

- **Automatización de pruebas.** Generación de rutinas de código (scripts) para agilizar y disminuir el tiempo de ejecución de pruebas.

B

- **Base de pruebas.** Es la información y/o documentación que se utilice para diseñar los casos y matrices de pruebas.

- **Benchmark.** Estándar contra el cual se pueden realizar comparaciones.

- **Bitácora de pruebas.** Registro cronológico de todos los detalles relevantes sobre la ejecución de una prueba.

- **Bloque.** Grupo de instrucciones continuas que son tratadas como una unidad.

- **Bug.** A nivel informático, es la manifestación de un error en el software. También es conocido como, error, fault, defecto, problema, incidente, anomalía, falla, entre otros.

C

- **Calidad** (*Quality*). Conjunto de propiedades y de características de un producto o servicio, que le confieren su aptitud para satisfacer unas necesidades explícitas e implícitas.

- **Capacidad de recuperación.** Característica de confiabilidad, que indica la capacidad del sistema para restablecer su nivel de respuesta después de un fallo crítico o error de hardware.

- **Capacidad de ser analizado.** Característica de mantenimiento, que indica la cantidad de esfuerzo requerido para diagnosticar la causa de un fallo.

- **Casos de prueba.** Conjunto de entradas, condiciones de ejecución y resultados esperados, desarrollados con el objetivo de probar algo concreto del software (ejecutar un camino del programa en particular, verificar la conformidad de un requisito concreto, detectar tipos de errores específicos).

- **Caso de uso.** Fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante ó esperado. Un caso de uso es en esencia, una interacción típica entre un usuario y un sistema de cómputo.

- **Checkpoint.** Punto en el cual, el estado, estatus o resultados de un programa han sido verificados o grabados.

- **Cobertura de decisión** (*decision coverage*). Numero de decisiones ejecutadas durante la ejecución de pruebas dividido entre numero total de decisiones en programa.

- **Cobertura de instrucción** (*statement coverage*). Numero de instrucciones ejecutadas durante la ejecución de pruebas dividido entre numero total de instrucciones en programa.

- **Cobertura de pruebas.** Grado en el cual un determinado grupo de pruebas planeadas para un sistema se logra ejecutar.

- **Coexistencia.** Característica de portabilidad, que indica la capacidad del software de coexistir con otro software independiente en un entorno común compartiendo recursos.

- **Complejidad.** (1)El grado de dificultad para entender el diseño o la implementación de un sistema o componente. (2)Pertinente a cualquier conjunto de métricas que midan el atributo en la definición anterior.

- **Componente.** Cada una de las partes que conforman a un sistema. Pueden ser subdivididos en otros componentes.

- **Comportamiento temporal** (*Time behavior*). Característica de eficiencia, que indica las características del software que influyen en el tiempo de respuesta y procesado y productividad cuando se ejecuta su función.

- **Comprensión** (*Understandability*). Característica de facilidad de uso, que indica las características del software que influyen en el esfuerzo del usuario para reconocer el concepto lógico y su aplicación.

- **Confiabilidad.** La habilidad de un sistema o componente de desempeñar su función requerida bajo condiciones especificas durante un periodo de tiempo determinado.

- **Control de calidad.** Se refiere a un grupo de actividades designadas a evaluar la calidad de los productos desarrollados o manufacturados. Nota: Este término no tiene un significado estandarizado en la ingeniería de software.

- **Controlador.** Módulo de software que llama, controla y monitorea la ejecución de uno o más módulos de software.

- **Controlador de pruebas.** Módulo de software usado para invocar al módulo probado y frecuentemente proveer de entradas para pruebas, controlar, monitorear la ejecución y reportar los resultados.
- **Criterio de aceptación.** Criterio que un sistema o componente debe satisfacer para que pueda ser aceptado por un usuario, cliente o cualquier otra entidad autorizada.

- **Criterio pasa/falla.** Reglas de decisión usadas para determinar si un elemento o característica del software pasa o falla una prueba.

- **Criterio de prueba.** El criterio que un sistema o componente debe cumplir para poder pasar una prueba determinada.

D

- **Defecto.** Una manifestación de un error.

- **Depuración.** Detectar, ubicar y corregir fallas en un programa de computadora. Conjunto de técnicas que incluye el uso de puntos de ruptura, pruebas de escritorio, vaciados, inspección, ejecución reversible, operación paso por paso y trazas.

- **Desarrollo del plan de proyecto.** Es tomar los resultados de los otros procesos de planeación y colocarlos un solo documento consistente y coherente.

- **Descripción de la prueba.** Véase “Especificación del caso de prueba”.

- **Detección de fallas.** Cualquier proceso que utiliza la ejecución de software como medio para determinar presencia o ausencia de fallas. Los tipos son pruebas y depuración.

- **Diccionario de fallas.** Lista de las fallas en un sistema o componente. Incluye también las pruebas diseñadas para su detección.

- **Diseño de pruebas.** Documento que especifica los detalles de la(s) prueba(s) que se va a realizar a una característica del software. Incluye el tipo de pruebas que se llevarán a cabo, así como los artefactos a utilizar para ello.

- **Disponibilidad.** El grado en el que un sistema o componente está accesible y operando cuando su uso es requerido.

- **Driver.** Es un programa que invoca un componente bajo pruebas, por ejemplo para simular un componente cuyo código todavía no está disponible (está todavía en desarrollo) o un componente externo.

E

- **Eficiencia** (*Efficiency*). Conjunto de características que determinan la relación entre el nivel de rendimiento del software y el número de recursos usados, bajo ciertas condiciones dadas. Se divide en las características comportamiento temporal y utilización de recursos.
- **Elemento de configuración.** Una agregación de hardware y/o software que está destinada a la administración de la configuración y es tratada como una sola entidad en el proceso de administración de la configuración.
- **Elemento de prueba.** Un módulo ó unidad de software que es el objeto de una prueba.
- **Error.** Anomalía dentro del software. Véase “Falla y anomalía”.
- **Error dinámico.** Un error que depende de la variación de una entrada conforme al tiempo.
- **Error nativo.** Error que es propio de la aplicación o sistema.
- **Esfuerzo de pruebas.** Proceso por el cual los Ingenieros de Pruebas de Software llevan a cabo la evaluación de un sistema/aplicación mediante pruebas conducidas, buscando detectar el mayor número de errores posibles para que sean removidos del sistema.
- **Especificación de requerimientos.** Documento generado durante la fase de elaboración o el análisis de requerimientos. También conocido como documento de análisis.
- **Especificación de casos de prueba.** Documento que detalla las entradas de las pruebas, condiciones de ejecución y resultados esperados para un elemento que va a ser probado.
- **Estabilidad** (*Stability*). Característica de mantenimiento, que indica volumen de riesgos de efectos inesperados tras una modificación.
- **Estimación** (*Estimate*). Es el resultado probable calculado, que regularmente se aplica a cuestiones cuantitativas como costos y lapsos de tiempo. Es el cálculo de la duración, del esfuerzo y/o del costo requerido para completar una tarea o un proyecto.
- **Evaluación de desempeño de la computadora.** Una disciplina de ingeniería que mide el desempeño de los sistemas computacionales e investiga métodos con los cuales puede ser mejorado el desempeño.

F

- **Facilidad de aprendizaje.** Característica de facilidad de uso, que indica las características software que influyen en el esfuerzo del usuario para aprender su aplicación (control, entrada, salida).

- **Facilidad de instalación.** Característica de portabilidad, que indica las características del software que influyen en el esfuerzo requerido para instalar el software en un entorno especificado.

- **Facilidad de prueba.** Característica de mantenimiento, que indica la capacidad del software para permitir que sea validado tras ser modificado.

- **Facilidad de uso.** Conjunto de características que influyen en el esfuerzo requerido para el uso y la evaluación individual de cada uso por parte de un conjunto de usuarios dados. Se divide en las características comprensión, facilidad de aprendizaje, operabilidad y atractivo.

- **Falla.** La incapacidad de un sistema o componente para realizar sus funciones requeridas conforme a requerimientos específicos. Defecto en el hardware causado por un error humano. Una desviación del funcionamiento esperado

- **Fase de instalación y revisión.** Periodo en el ciclo de vida de un sistema que está integrado y probado en el entorno de operación para asegurarse de que funciona conforme a lo requerido.

- **Funcionalidad.** Grado en que las necesidades asumidas o descritas se satisfacen.

G

- **Generador de casos de prueba.** Herramienta de software que toma como entrada código fuente, criterios de prueba, especificaciones y/o definiciones de estructuras de datos. Utiliza esta información para generar datos de entrada para pruebas y en algunos casos determina los resultados esperados.

- **Ghost.** Aplicación con la cual se realiza el copiado del contenido del disco duro de una computadora en un solo archivo comprimido o conjunto de archivos (llamado: imagen), para que así puedan copiarse dichos contenidos (incluso la información de la configuración y aplicaciones) al disco duro de otras computadoras.

- **Guía paso a paso.** Técnica de análisis estático en la que un diseñador o programador lleva a los miembros de un equipo de desarrollo y demás partes interesadas a través de un segmento de documentación o de código y los participantes hacen preguntas y comentarios sobre posibles errores, violación de estándares y otros problemas.

H

- **Happy path**. Probar el recorrido del flujo básico en una aplicación.

I

- **Incidente de prueba de software.** Cualquier evento que ocurre durante la ejecución de una prueba de software y requiere cierto grado de investigación.
- **Inspección.** Técnica de análisis estático que depende del examen visual del desarrollo para la detección de errores, incumplimiento de estándares y otros problemas. Los diferentes tipos incluyen inspección de diseño e inspección de código.
- **Inspección de pruebas.** Proceso formal en el que las pruebas son inspeccionadas en busca de defectos.
- **Instrumentación.** Dispositivos o instrucciones insertadas en el hardware o en el software para monitorear la operación de un sistema o componente.
- **Interfase gráfica de usuario (GUI).** Interfase entre la computadora y el usuario, donde este último puede manipular objetos y realizar tareas.
- **Inventario de pruebas.** Lista completa de todas las pruebas conocidas, rutas, datos, módulos, diseño, sistemas, etc., que se tienen planeadas para realizar a un producto de software.
- **Ingeniero de pruebas (IP)**. También conocido como "Tester".
- **Iterativo incremental.** Atacar primero las partes del sistema cargadas de riesgo para buscar estabilidad, y completando después las partes más rutinarias en iteraciones sucesivas, cada una de las cuales lleva un incremento del progreso hasta la versión final.

J

-

K

-

L

-

M

- **Mantenimiento.** Esfuerzo requerido para implementar cambios. Se divide en las características capacidad de ser analizado, cambiabilidad, estabilidad y facilidad de prueba.

- **Mapas de pruebas.** Un mapa de pruebas también conocido como mapas de transición de estados o grafos, son una representación gráfica de los requerimientos funcionales a probar.

- **Máquina virtual.** Permite simular el hardware de una PC completa real en una misma máquina, pudiendo accederlas de manera fácil y rápida, un ejemplo de uso es para poder aplicar pruebas y representar más ambientes de prueba en una sola máquina.

- **Matriz de prueba.** Es una combinatoria de todas las variantes de los casos de prueba con cada uno de sus valores, de tal forma que no haya algún escenario sin validar por más remoto o poco probable que suceda.

- **Método de aserción inductiva.** Técnica que prueba el grado de "correctividad" en el que se emplean aserciones para describir las entradas, salidas y condiciones intermedias. Un conjunto de teoremas son desarrollados relacionando la satisfacción de las aserciones de entrada con las de salida y evaluados mediante inducción.

- **Métrica fundamental.** Medida de una magnitud física donde el nombre medido corresponde con el de la métrica. Es la métrica más simple que se puede generar. Por ejemplo, errores por cada 100 líneas de código.

- **Modelo de error.** Modelo utilizado para estimar o predecir el número de errores, tiempo requerido de prueba y características similares de un sistema.

- **Modelo de fallas.** Identifica relaciones y componentes del sistema bajo pruebas que son más probables a presentar fallas.

- **Módulo.** Unidad de programa que es discreta e identificable con respecto a la compilación.

- **Mutación de programa.** Un programa que ha sido alterado con el fin de evaluar la habilidad de los casos de prueba para detectar la alteración.

N

-

Ñ

-

O

- **Operabilidad.** Característica de facilidad de uso, que indica las características del software que influyen en el esfuerzo del usuario para operar y control operacional.

- **Outsourcing.** Subcontrata de las partes de procesos para que sean realizados por empresas externas.

P

- **Partición.** Sección del disco duro que actúa como un elemento independiente, es decir, como si fuera otra máquina.

- **Plan del proyecto** (*project plan*). Es un documento formalmente aprobado, utilizado como guía tanto para la ejecución como para el control del proyecto, Los usos principales del plan del proyecto son documentar los supuestos y decisiones planificadas, facilitar la comunicación entre las entidades involucradas en el proyecto, y documentar las bases aprobadas de alcance, costos y programa. Un plan del proyecto puede presentarse de forma resumida o detallada.

- **Plan de pruebas.** Documento que describe el enfoque, alcance, recursos e itinerarios de las actividades necesarias para pruebas. Identifica los elementos de prueba, las características a ser evaluadas, tareas que tienen que ser realizadas, el responsable de realizar cada tarea, los recursos necesarios y cualquier riesgo que requiera un plan de contingencia.

- **Portabilidad.** Conjunto de características que determinan la capacidad del software para ser transferido de un entorno de operación a otro.

- **Precisión.** Característica de funcionalidad, que indica el grado de exactitud de los efectos del sistema (salida).

- **Predicción de errores.** Una declaración cuantitativa sobre el número y naturaleza de las fallas esperadas en un sistema o componente.

- **Prevención de fallas.** Cualquier proceso usado para reducir el potencial de aparición de fallas en el software. Las formas de hacerlo son: administración de la configuración, revisiones y metodologías de desarrollo.

- **Probar.** (1)Proceso de operar un sistema o componente en condiciones específicas observando, registrando y evaluando los resultados. (2)El proceso de analizar un elemento de software, ya sea de manera manual o automatizada, para detectar las diferencias entre las condiciones existentes y las requeridas (es decir, errores) y evaluar sus características.

- **Procedimiento de prueba.** Instrucciones detalladas acerca de la preparación, ejecución y evaluación de los resultados para un caso de prueba o requerimiento funcional determinado.

- **Prueba.** Actividad en la que un sistema o componente es ejecutado bajo condiciones específicas y los resultados son registrados y evaluados.

- **Pruebas de aceptación.** Prueba formal para determinar si el sistema satisface los criterios de aceptación, y permitir al cliente que determine si acepta o no acepta el sistema.

- **Prueba alfa.** Prueba simulada o prueba de operación actual, utilizando una liberación alfa del sistema, desarrollada con un equipo selecto de usuarios que reporta errores y otras observaciones que se les informan a los desarrolladores.

- **Prueba beta.** Prueba que se realiza después de que se completa la prueba alfa. Esta prueba se ejecuta con un conjunto mayor de usuarios en un versión beta del sistema y antes de que el mismo sea liberado a todos los usuarios.

- **Prueba de caja blanca.** Se determinan cuáles son los casos de prueba a partir del código fuente del software y se utilizan las especificaciones para determinar el resultado esperado del caso

- **Prueba de caja negra.** Realizadas directamente sobre la interfaz de usuario manipulando los datos de entrada y revisando los resultados. Están basadas en los requerimientos y la funcionalidad, no se necesitan conocimientos del diseño interno o del código.

- **Prueba de componentes.** Pruebas realizadas a componentes individualmente o por grupos.

- **Prueba de comportamiento.** Desarrollo de los casos de pruebas se basa en las funcionalidades y/o el comportamiento que el software tiene que tener (requisitos, especificaciones, conocimiento del dominio, repositorio de defectos, etc.).

- **Prueba conductual o de comportamiento.** Prueba que ignora el mecanismo interno de un sistema y se enfoca únicamente a las salidas generadas como respuesta a entradas específicas y condiciones de ejecución determinadas.

- **Prueba de configuración.** Son pruebas orientadas a diversas configuraciones de hardware y software. Se realizan pruebas bajo diferentes configuraciones de:

    - Hardware: discos duros, impresoras, CPU, tarjetas gráficas, tarjetas de sonido.
    - Sistemas operativos y/o versiones de un sistema operativo.
    - Sistemas GUI (MS Windows, X-Windos) y sus diferentes versiones.
    - Bases de datos.

- **Prueba de correctividad.** Técnica formal usada para probar matemáticamente que un programa de computadora cumple con los requerimientos especificados.

- **Prueba de datos** (integridad de la información). Verifica que los datos sean consistentes en la base de datos es decir, valida que todos los movimientos realizados en el sistema estén reflejados en la base de datos de manera correcta.

- **Prueba de desarrollo.** Prueba que puede ser formal o informal y realizada durante el desarrollo de un sistema o componente. Generalmente la lleva a acabo el desarrollador en el entorno de desarrollo.

- **Prueba de desempeño (pruebas de performance).** Prueba realizada que evalúa el apego de un sistema o componente a los requerimientos de desempeño. Los tipos incluyen estrés y volumen.

- **Prueba dinámica.** Pruebas realizadas con el programa en ejecución tal y como lo utilizaría el usuario, ejecutando casos de prueba y matrices

- **Prueba de escritorio.** Técnica de análisis estático en la que el listado de un código, resultado de pruebas u otra documentación son examinadas visualmente, usualmente por el mismo creador, para identificar errores y problemas.

- **Prueba estadística.** Su finalidad es el predecir el grado de calidad del software. Se usa en la forma más pura de la ingeniería del software y existen diferentes técnicas disponibles.

- **Prueba estática.** Es aquella que se realiza cuando el programa no está en ejecución, únicamente se examina y se revisan aspectos como estandarización de pantallas, orden de campos, documentación

- **Prueba de estrés.** Prueba realizada para evaluar a un sistema o componente más allá de los límites de los requerimientos especificados. Véase Prueba de Desempeño.

- **Pruebas formales.** Pruebas llevadas a cabo de acuerdo a planes y procedimientos que han sido revisados y aprobados por un cliente, usuario o el nivel administrativo correspondiente.

- **Prueba funcional.** Una prueba de las funciones de un programa, es decir, que cumpla con todas las funciones que el usuario especificó.

- **Prueba de flujo de datos.** Prueba que se ocupa de las variables, definición y uso dentro de un programa.

- **Prueba informal.** Prueba realizada de acuerdo a planes y procedimientos de pruebas que no han sido revisados y aprobados por el cliente/usuario o el nivel administrativo correspondiente.

- **Pruebas de instalación / desinstalación.** Verifican que el setup de la aplicación funcione correctamente, que se instalen completamente cada uno de los componentes del mismo y que de la misma forma se realice la desinstalación.

- **Prueba de integración.** Prueba en la que los componentes de software y/o hardware son combinados y probados para evaluar las interacciones que se dan entre ellos.

- **Prueba de interfase.** Prueba llevada a cabo para evaluar si el sistema transfiere los datos y el control correctamente.
- **Puntos de función.** Una métrica sintética del software que se compone de totales ponderados de las entradas, salidas, archivos lógicos o grupos de datos de usuario e interfases correspondientes a una aplicación.

- **Prueba de recuperación.** Tipo de prueba que involucra interrupciones intencionales a un sistema para asegurar que los métodos de recuperación de datos funcionan correctamente.

- **Prueba de regresión.** Consiste en rehacer pruebas seleccionadas a un componente o sistema para verificar que las modificaciones hechas no hayan causado efectos indeseables o que todavía están en apego a los requerimientos especificados.

- **Prueba de ruta.** Prueba diseñada para ejecutar todos los caminos en un programa. Técnica que permite derivar una estructura de flujo de un diseño o código y usar esta estructura como una guía para definir un conjunto básico de casos de pruebas (caminos de ejecución).

- **Pruebas de seguridad.** Pruebas centradas en asegurar que los datos (o sistemas) son accesibles solamente a los usuarios previstos por el cliente.

- **Prueba de sentencia.** Prueba diseñada para ejecutar cada sentencia en un programa.
- **Prueba de sistema.** Prueba realizada en un sistema completo que evalúa su apego a los requerimientos especificados.
- **Pruebas de software.** Examinar un artefacto de software con la intención de encontrar defectos (tal que tus clientes no lo hagan).
- **Prueba unitaria.** Pruebas realizadas a unidades de software; estas pruebas son realizadas por los desarrolladores.
- **Pruebas de usabilidad.** Permiten verificar que la aplicación sea lo suficientemente amigable con el usuario final
- **Prueba de volumen.** Prueba que simula una gran cantidad de entradas bajo una carga normal del sistema. Usualmente se asocia con sistemas de procesamiento por lotes.
- **Punto de revisión.** Véase Checkpoint.

Q

-

R

- **Reemplazo.** Característica de portabilidad, que indica las características del software que influyen en la posibilidad y esfuerzo requerido para usarlo en lugar de otro software en el mismo entorno.
- **Repetibilidad de la prueba.** Un atributo de la prueba que indica que se obtienen los mismos resultados cada vez que la prueba es ejecutada.
- **Reporte de incidente de prueba.** Documento que describe un evento que ha ocurrido durante las pruebas y requiere cierto grado de documentación.
- **Reporte de prueba.** Documento que describe la realización y resultados de las pruebas de un componente o sistema.
- **Revisión.** (1)Prueba llevada a cabo en el entorno operacional o de mantenimiento para asegurarse de la correcta ejecución de un software después de su instalación. (2)Proceso en el cual los resultados del trabajo son presentados al personal del proyecto, administradores, usuarios, clientes y demás entidades interesadas para su discusión y aprobación. Los tipos incluyen revisión de código, revisión de diseño, revisión de requerimientos.
- **Revisión crítica de sistema.** Revisión que se hace para: verificar que el diseño a detalle de uno o más elementos de configuración satisfagan los requerimientos especificados; establecer la compatibilidad entre los elementos de configuración y demás dispositivos, software y personal; evaluar áreas de riesgo para cada elemento; y, si aplica, evaluar los

resultados de los análisis de productividad, revisar las especificaciones del hardware, evaluar planes de pruebas preeliminares y evaluar el nivel de adecuación de los documentos preeliminares sobre operación y soporte.

- **Revisión del diseño del sistema.** Revisión que se lleva a cabo para evaluar la forma en la que los requerimientos de un sistema han sido llevados a los elementos de configuración, el proceso de ingeniería de software que se llevo a cabo para acomodar los requerimientos, la planeación para la siguiente etapa, consideraciones de manufactura y la planeación para la producción.

- **Revisión preeliminar de diseño.** Revisión llevada a cabo para verificar el progreso, adecuación tecnológica y resolución de riesgos de un enfoque de diseño seleccionado para uno o más elementos de configuración; para determinar la compatibilidad de cada diseño con los requerimientos para el elemento de configuración; para evaluar el grado de definición y estimar el riesgo asociado con los procesos y métodos de manufactura seleccionados; para establecer la existencia y compatibilidad de las interfases físicas y funcionales entre los elementos de configuración y otros dispositivos, software y personal; y para evaluar los documentos preeliminares de operación y soporte.

- **Revisión de la preparación de la prueba.** Revisión que se lleva a cabo para evaluar resultados preeliminares de la prueba para uno o más elementos de configuración; para verificar que los procedimientos de prueba de cada elemento de configuración están completos, se apegan a los planes de prueba y satisface los requerimientos de la prueba; y para verificar que el proyecto está preparado para la prueba formal de los elementos de configuración.

- **Riesgo.** Un riesgo se define como cualquier evento o condición que pueda tener un impacto negativo o positivo en los resultados de un proyecto.

- **Ruta.** Secuencia de instrucciones que pueden ser llevadas a cabo durante la ejecución de un programa.

S

- **Scripts de prueba.** Conjunto de instrucciones entendibles por la computadora que automatizan la ejecución de un método de prueba (o de la porción de un método de prueba) y se usan para implementar y ejecutar un caso de pruebas de forma eficiente y efectiva.

- **Seguimiento** (*monitoring*). Consiste en la toma de datos, su análisis, y posteriormente informe sobre la realización del proyecto, generalmente comparándolo con el plan.

- **Seguridad.** Característica de funcionalidad, que indica el grado en que un acceso no autorizado (accidental o deliberado) se prevenga y se permita un acceso autorizado.

- **Sistema de verificación automatizada.** Una herramienta de software que acepta como entrada un programa de computadora, una representación de los procesos y, posiblemente con ayuda humana, una aprobación o desaprobación del mismo.

- **Software Test Engineer (STE).** Persona conocida también como ingeniero de pruebas de software, o tester. Es aquél que tiene como labor ejecutar las pruebas de un proyecto.

- **Software Test Engineer Senior (STE Sr.).** Ingeniero de pruebas de software con más experiencia en el área de pruebas; podrá apoyar a su líder en otras actividades, a parte de la ejecución de pruebas.

- **Software Test Lead (STL).** Líder de un equipo de ingenieros de pruebas de software (STEs y STEs Sr.).

- **Software Test Manager (STM).** Administrador de pruebas de software. Organiza tanto al líder de pruebas (STL), como al equipo de ingenieros de prueba (STEs y STEs Sr.).

- **Stakeholder.** Cualquier persona interesada en, afectada por y/o implicada con el funcionamiento del sistema software. Por ejemplo, el usuario, el cliente, nuestra empresa, etc.

- **Stub.** Implementación de un propósito específico para un módulo de software. Se usa para desarrollar un módulo de pruebas al que llama o es dependiente del stub de otra forma.

T

- **Tabla de decisión.** Tabla usada para mostrar el conjunto de condiciones y los resultados de las mismas.

- **Test case point.** Técnica de estimación del esfuerzo aplicado a la realización de pruebas de un sistema o componente que se basa en la ponderación de los elementos del caso, como son: los actores, transacciones, nivel de complejidad, etc.

- **Test Management Approach (TMAP)**. Metodología que abarca todos los aspectos concernientes al proceso de pruebas de software desde gestión de alto nivel a detalles del uso de técnicas TMAP.

- **Test Maturity Model (TMM)**. Modelo del Software Engineering Institute (SEI) para la evaluación y mejora del proceso de test en una organización.

- **Test Process Improvement (TPI)**. Modelo que proporciona una idea general de la madurez del proceso de pruebas en una organización, a partir de ahí se establecen unos pasos de mejora graduales y controlados.

- **Test to pass.** Asegurar que los procesos fundamentales (mínimos) del software funcionan correctamente. (Happy Path).

- **Test to fail.** Diseñar y ejecutar pruebas con el propósito de corromper el software.

- **Testware.** Todo producto relacionado con las pruebas de software; desde un proceso de pruebas establecido, experiencia, automatización, plan de pruebas, casos de prueba, hasta aplicaciones y herramientas de pruebas.

- **Tiempo transcurrido entre fallas.** El tiempo esperado u observado que transcurre entre fallas consecutivas en un sistema o componente.

- **Tiempo transcurrido para reparación.** El tiempo esperado u observado que es requerido para reparar un sistema o componente y reanudar su operación normal.

- **Time to market.** Es el tiempo que transcurre entre que un producto es concebido y está disponible para la venta.

- **Tolerancia a fallas.** La habilidad de un sistema o componente para continuar su operación normal pese a la presencia de fallas en el hardware y/o software.

U

- **Unidad.** Elemento especificado en el diseño del software que se prueba por separado.

- **Usabilidad.** La facilidad con la que un usuario puede aprender a operar, preparar entradas y obtener salidas de un sistema o componente.

- **Utilización de recursos.** Característica de eficiencia, que indica las características del software que influyen en el número de recursos usados, y la duración de su uso, cuando se lleva a cabo su función.

V

- **Validación.** Inspeccionar los productos finales para garantizar el cumplimiento de los requerimientos.

- **Validación y verificación independiente.** Verificación y validación llevada a cabo por una organización que es tecnológica, administrativa y financieramente independiente de aquella que hizo el desarrollo (outsourcing).

- **Valor límite o frontera.** Un valor que corresponde con el máximo o mínimo permisible a una entrada, dato interno o salida de un sistema o componente.

- **Verificación.** Inspeccionar productos intermedios y el proceso de desarrollo. Es asegurar que el software esté libre de defectos.

- **Verificación del alcance** (*scope verification*). Consiste en asegurar que todas las entregas identificadas del proyecto se han terminado satisfactoriamente.

W

- **Walkthrough** Una revisión que es menos formal que las inspecciones. El líder se llama coordinador y es el autor, presentador, secretario. No hay checklists como en las inspecciones. Walkthroughs se utilizan más para diseños y código. Durante la reunión los participantes van paso a paso a través del (pseudo) código como un tipo de ejecución manual del código donde los participantes simulan un ordenador.

X

-

Y

-

Z

-

BIBLIOGRAFÍA

- Koomen, T., Pol, M.: "**Test Process Improvement: a Practical Step-by-Step Guide to Structured Testing**", editorial Addison Wesley Longman, 1999, ISBN 0-201-59624-5.

- Presuman, R.S.: "**Ingeniería del Software: un Enfoque Práctico**", 6ª Edición, editorial MacGraw-Hill, 2006.

- Piattelli Palmarinini, M.G.: "**Calidad en el Desarrollo y Mantenimiento del Software**", 4ª Edición, editorial Ra-Ma, 2002.

- Pressman, A.H.: "**Ingeniería del Software**", 4ª Edición, editorial McGraw-Hill, 2000.

- Project Management Institute, Inc., "**Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK®)**", Tercera Edición, 2004

- Software Program Managers Network, "**Little Book of Testing – Volume I: Overview and Best Practices**", SPMN, June 1998

- Software Program Managers Network, "**Little Book of Testing – Volume II: Implementation Techniques**", SPMN, June 1998

REFERENCIAS

- ANSI/IEEE Standard 829-1983, "**IEEE Standard for Software Test Documentation**". IEEE Press, 1983. Este estándar se incluye en la colección *Software Engineering Standards* publicado por IEEE Press.

- IEEE Standard 829-1998, "**IEEE Standard for Software Test Documentation**", The Institute of Electrical and Electronics Engineers, 1998.

- Brian Marick, "**New Models for Test Development**", Proceedings of International Quality Week, May 1998.

- Spillner, A.: "**The W-Model – Strengthering the Bond Between Development and Test**", Proceedings Conquest - 4th Conference on Quality Engineering in Software Technology, 2000.

- The Stationery Office. **Managing Successful Projects with PRINCE2**. ISBN 0-11-330891-4 (Official PRINCE2 publication).