1 Fundamentos del Testing

- ✓ **La importancia económica del software**

El funcionamiento de maquinaria y equipamiento depende en gran medida del software.
No es posible imaginar grandes sistemas, en el ámbito de las finanzas ni el control del tráfico automotor, entre otros, funcionando sin software.

- ✓ **Calidad del Software**

Cada vez más, la calidad software se ha convertido en un factor determinante del éxito de sistemas y productos técnicos o comerciales.

- ✓ **Pruebas para la mejora de la calidad**

Las pruebas y revisiones aseguran la mejora de la calidad de productos de software así como de la calidad del proceso de desarrollo en sí.

Riesgo
No todo el software tiene el mismo nivel de riesgo y no todos los problemas tienen el mismo impacto cuando ocurren.

1.1 Calidad de Software

La calidad está constituida por:

- ✓ Atributos funcionales de calidad:

  Funcionalidad: correctitud y completitud de los requisitos del usuario.

- ✓ Atributos NO funcionales de calidad:

  Fiabilidad: el sistema mantendrá su capacidad y funcionalidad a lo largo de un período de tiempo.
  Usabilidad: fácil de usar, fácil de aprender, conforme a normas y uso intuitivo.
  Portabilidad: fácil de instalar y desinstalar, y configurar parámetros.

1.2 Calidad

Grado en el cual un componente, sistema o proceso satisface requisitos especificados y/o necesidades y expectativas del usuario/cliente.

Rol del Testing en el desarrollo, mantenimiento y operaciones de software (K2)
La prueba rigurosa de sistemas y la documentación pueden ayudar a reducir el riesgo de que ocurran problemas en un entorno operacional y a contribuir a la calidad del sistema

de software, si los defectos encontrados son corregidos antes que el sistema sea lanzado para uso operacional.
La prueba de software puede ser también requerida para cumplir con requisitos legales o contractuales o con estándares específicos de la industria.

1. Caso de Prueba (Test Case/Test Basis)
2. Código (source code)
3. Depuración (debugging)
4. Desarrollo de Software (Software development)
5. Requisitos (requirement)
6. Revisión (Review)

1. Caso de prueba (Según IEEE std 610)

- Precondiciones
- Conjunto de Valores de entrada
- Conjunto de resultados esperados
- Forma en la cual se debe ejecutar el caso de prueba y verificar los resultados
- PosCondiciones esperadas
  Conjunto de documentos que definen los requisitos de un componente o sistema.

2. Código (source code) Según IEE 610

Instrucciones de ordenador y definiciones de datos expresados en un lenguaje de programación o en una forma de salida generada por un ensamblador, compilador u traductor

3. Depuración (debugging)
Proceso de encontrar, analizar y eliminar las causas de los fallos en el software
Localización y corrección de defectos en el código fuente

4. Desarrollo de Software (Software development)
Proceso/secuencia de actividades cuyo objetivo es desarrollar un sistema basado en un computador

5. Requisito (requirement) Según IEEE 610
Condición o capacidad necesaria para un usuario con el objeto de solucionar un problema o lograr un objetivo que debe ser alcanzado o poseído por un sistema o componente de un sistema, para satisfacer un contrato, estándar, especificación, u otro documento impuesto formalmente

6. Revisión (Review) Según IEEE 1028
Evaluación de un producto o del estado de un proyecto para detectar discrepancias con los resultados planificados y para recomendar mejoras.

1.3 ¿Por qué es necesaria la prueba?

1.3.1 Términos

Bug, defecto, error, falla, avería, calidad, riesgo, software, prueba.

1.3.2 Contexto de los sistemas de software

Los sistemas de software son una parte creciente de la vida, desde las aplicaciones de negocio (ej. transacciones bancarias) hasta productos de consumo (ej. autos). La mayoría de la gente ha tenido una experiencia con software que no funcionó como se esperaba. El software que no funciona correctamente puede conducir a muchos problemas, incluyendo pérdida de dinero, tiempo o reputación del negocio y podría incluso causar lesión o muerte.

1.3.3 Causas de defectos de software

Un ser humano puede hacer un error, el cual produce un defecto (avería, bug) en el código, en el software o en un sistema, o en un documento. Si un defecto en el código es ejecutado, el sistema fallará en hacer lo que debía hacer (o hacer algo que no debía), causando una falla. Los defectos en el software, sistemas o documentos pueden resultar en fallas, mas no todos los defectos hacen eso.

Los defectos ocurren porque los seres humanos son falibles y porque hay presión de tiempo, código complejo, complejidad de infraestructura, tecnologías cambiadas y/o muchas interacciones del sistema.

Las fallas pueden ser causadas por condiciones ambientales asimismo: la radiación, magnetismo, campos electrónicos y polución pueden causar averías en el soporte lógico incorporado o influenciar la ejecución del software al cambiar las condiciones del hardware.

1.3.4 Rol de la prueba en el desarrollo, mantenimiento y operaciones de software

La prueba rigurosa de sistemas y la documentación pueden ayudar a reducir el riesgo de que ocurran problemas en un entorno operacional y a contribuir a la calidad del sistema de software, si los defectos encontrados son corregidos antes que el sistema sea lanzado para uso operacional.

La prueba de software puede ser también requerida para cumplir con requisitos legales o contractuales o con estándares específicos de la industria.

1.3.5 Prueba y calidad

Con la ayuda de la prueba, es posible medir la calidad del software en términos de defectos encontrados, para requisitos y características de software tanto funcionales como no funcionales

(ej. fiabilidad, usabilidad, eficiencia, mantenibilidad). Para mayor información sobre la prueba no funcional vea el capítulo 2; para mayor información sobre características de software vea "Ingeniería de Software – Calidad del Producto de Software" (ISO 9126).

La prueba puede dar confianza en la calidad del software si encuentra pocos o ningún defecto. Una prueba diseñada apropiadamente que pase, reduce el nivel total de riesgo de un sistema. Cuando la prueba encuentra defectos, la calidad del sistema de software se incrementa cuando defectos son arreglados.

Las lecciones deberían aprenderse de proyectos anteriores. Al entender las causas raíz de los defectos encontrados en otros proyectos, los procesos pueden ser mejorados, lo cual a su vez debería prevenir que aquellos defectos vuelvan a ocurrir y, como una consecuencia, mejorar la calidad de futuros sistemas.

La prueba debería estar integrada como una de las actividades de garantía de calidad (es decir, junto con estándares de desarrollo, entrenamiento y análisis de defectos).

1.3.6 ¿Cuánta prueba es suficiente?

El decidir cuánta prueba es suficiente debería tomar cuenta del nivel de riesgo, incluyendo productos técnicos y del negocio y riesgos del proyecto, y las restricciones del proyecto tales como tiempo y presupuesto.

La prueba debería proporcionar suficiente información a las partes interesadas para realizar decisiones informadas sobre el lanzamiento del software o sistema que está siendo probado, para el próximo paso de desarrollo o entrega a los clientes.

1.4 ¿Qué es prueba?

1.4.1 Términos

Código, depuración, desarrollo (de software), requisito, revisión, base de pruebas, caso de prueba, prueba, objetivos de prueba.

1.4.2 Introducción

Una percepción común de la prueba consiste en que solamente consiste de realizar pruebas, es decir ejecutar el software. Esto es parte de la prueba, más no todas las actividades de prueba.

Las actividades de la prueba existen antes y después de la ejecución de la prueba, las actividades tales como planificación y control, eligiendo condiciones de prueba, diseñando casos de la prueba y comprobando los resultados, evaluando criterios de finalización, informando sobre el proceso de prueba y el sistema bajo prueba, y finalizando o terminación (ej. después de que una fase de

prueba ha sido completada). La prueba también incluye la revisión de documentos (incluyendo el código fuente) y análisis estático.

Tanto como la prueba dinámica y la prueba estática pueden ser utilizadas como un medio para alcanzar objetivos similares y proporcionarán la información para mejorar tanto el sistema que se probará como los procesos de desarrollo y de prueba.

Puede haber diferentes objetivos de prueba:

- ✓ encontrar defectos.
- ✓ ganar confianza sobre el nivel de calidad y proporcionar información.
- ✓ prevenir defectos.

El proceso de pensamiento de diseñar pruebas temprano en el ciclo de vida (que verifica la base de prueba vía el diseño de prueba) puede ayudar a evitar que los defectos sean introducidos en el código. Las revisiones de documentos (ej. los requisitos) también ayudan a prevenir los defectos que aparecen en el código.

Los diversos puntos de vista en la prueba toman en cuenta diferentes objetivos. Por ejemplo, en las pruebas de desarrollo (ej. Pruebas de componente, de integración y de sistema), el objetivo principal puede ser causar cuantas fallas como sea posible para que los defectos en el software sean identificados y puedan ser arreglados. En las pruebas de aceptación, el objetivo principal puede ser confirmar que el sistema trabaja según lo esperado, para ganar confianza de que cumple los requisitos. En algunos casos el objetivo principal de la prueba puede ser evaluar la calidad del software (sin la intención de arreglar los defectos), para dar información a las partes interesadas del riesgo de lanzar el sistema en un momento dado. Las pruebas de mantenimiento a menudo incluyen pruebas de que ningún error nuevo ha sido introducido durante el desarrollo de los cambios. Durante las pruebas operacionales, el objetivo principal puede ser evaluar las características del sistema tales como fiabilidad o disponibilidad.

La depuración y la prueba son diferentes. La prueba puede mostrar las fallas que son causadas por los defectos. La depuración es la actividad de desarrollo que identifica la causa de un defecto, repara el código y comprueba que el defecto haya estado arreglado correctamente. Las pruebas de confirmación subsiguientes por parte de un probador garantizan que el arreglo resuelve en sí la falla. La responsabilidad para cada actividad es muy diferente, es decir, los probadores prueban y los desarrolladores depuran.

1.5 Principios generales de prueba

1.5.1 Términos

Prueba exhaustiva.

1.5.2 Principios

Un número de principios de prueba se han sugerido durante los últimos 40 años y ofrecen pautas generales comunes para toda prueba.

1.5.2.1 Principio 1 - La prueba muestra la presencia de defectos

La prueba puede mostrar que los defectos están presentes, pero no puede probar que no hay defectos. La prueba reduce la probabilidad de los defectos no descubiertos restantes en el software pero, incluso si no se encuentran defectos, no es una prueba de corrección.

1.5.2.2 Principio 2 - La prueba exhaustiva es imposible

Probar todo (todas las combinaciones de entradas y de precondiciones) no es factible a excepción de trivial casos. En vez de la prueba exhaustiva, usamos el riesgo y las prioridades para enfocar los esfuerzos de prueba.

1.5.2.3 Principio 3 - Prueba temprana

Las actividades de prueba deberían comenzar tan pronto como sea posible en el ciclo de vida del desarrollo del software o del sistema y deberían estar enfocados en los objetivos definidos.

1.5.2.4 Principio 4 - Agrupamiento de defectos

Un número pequeño de módulos contiene la mayoría de los defectos descubiertos durante la prueba de pre-lanzamiento o mostrar las fallas más operacionales.

1.5.2.5 Principio 5 - Paradoja del pesticida

Si las mismas pruebas se repiten una y otra vez, eventualmente el mismo conjunto de casos de prueba no encontrará más cualquier nuevo bug. Para superar esta "paradoja del pesticida", los casos de prueba necesitan ser repasadas y revisadas regularmente, y nuevas y diversas necesitan ser escritas al ejercicio diferente partes del software o del sistema potencialmente para encontrar más defectos.

1.5.2.6 Principio 6 - La prueba es dependiente del contexto

La prueba se hace diferentemente en diversos contextos. Por ejemplo, se prueba el software de seguridad crítica diferentemente de un sitio de e-comercio.

1.5.2.7 Principio 7 - Falacia de la ausencia de errores

Encontrar y arreglar defectos no ayuda si el sistema construido es inutilizable y no cumple las necesidades y expectativas de los usuarios.

1.6 Proceso fundamental de prueba

1.6.1 Términos

Pruebas de confirmación, criterios de salida, incidente, pruebas de regresión, base de prueba, condición de prueba, cobertura de prueba, datos de prueba, datos de prueba, ejecución de prueba, registro de prueba, estrategia de prueba, informe de resumen de la prueba, testware.

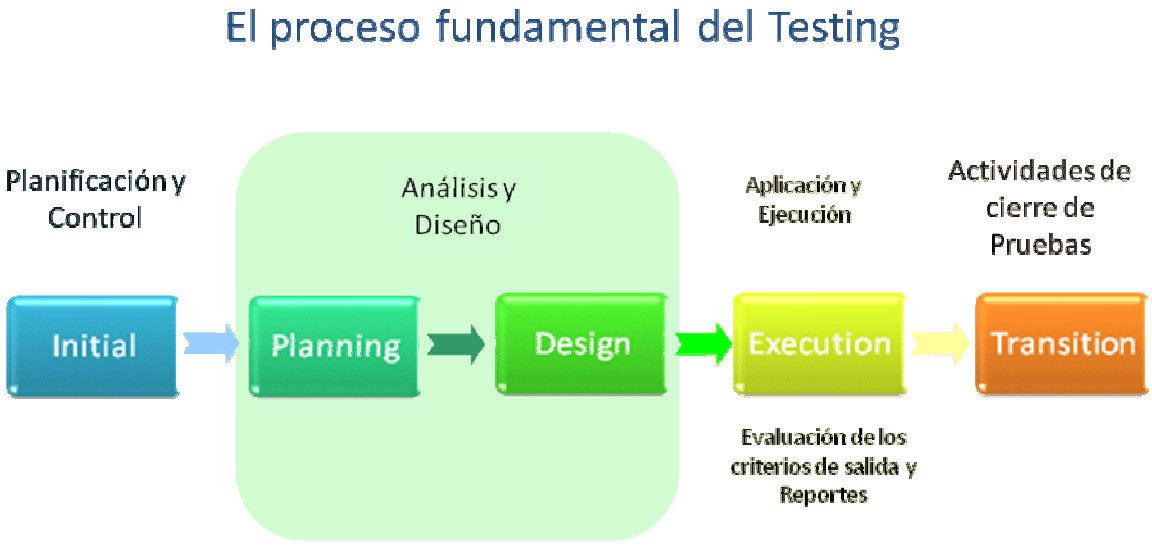
1.6.2 Introducción

La parte más visible de las pruebas es ejecutar las pruebas. Pero para ser eficaces y eficientes, los planes de prueba deben también incluir el tiempo que se tomará al planificar las pruebas, diseñar los casos de prueba, prepararse para la ejecución y evaluar el estado.

El proceso de prueba fundamental consiste de las siguientes actividades principales:

- ✓ planificación y control;
- ✓ análisis y diseño;
- ✓ implementación y ejecución;
- ✓ evaluar criterios salida e informar;
- ✓ actividades de cierre de prueba.

Aunque lógicamente secuenciales, las actividades en el proceso pueden superponerse u ocurrir concurrentemente.



La parte más visible de las pruebas es la ejecución. Pero para ser eficaz y eficiente, los Test Plans deben incluir también el tiempo para gastar en la planificación de las pruebas, el diseño de casos de prueba, la preparación para su ejecución y evaluar su estado.

1.6.3 Panificación y control de prueba

La planificación de la prueba es la actividad de verificar la misión de la prueba, definiendo los objetivos de la prueba y la especificación de las actividades de la prueba para satisfacer los objetivos y la misión.

El control de la prueba es la actividad en curso de comparar el progreso real contra el plan, y de informar el estado, incluyendo las desviaciones del plan. Implica tomar las acciones necesarias para cumplir la misión y objetivos del proyecto. Para controlar la prueba, ésta debe ser supervisada a través del proyecto. La planificación de la prueba considera la información del monitoreo y control de actividades.

Las tareas de planificación de prueba tienen las siguientes tareas principales:

- ✓ Determinar el alcance y los riesgos, e identificar los objetivos de la prueba.
- ✓ Determinar el acercamiento de prueba (técnicas, elementos de prueba, cobertura, identificar y realizar interfaz con los equipos involucrados en las pruebas, testware).
- ✓ Determinar los recursos de prueba requeridos (por ejemplo gente, entorno de prueba, PCs).
- ✓ Implementar la política de prueba y/o estrategia de prueba.
- ✓ Programar los análisis de prueba y las tareas de diseño.
- ✓ Programar la implementación, ejecución y evaluación de prueba.
- ✓ Determinar los criterios de salida.

Las tareas de control de prueba tienen las siguientes tareas principales:

- ✓ Medir y analizar los resultados.
- ✓ Monitorear y documentar el progreso, la cobertura de prueba y los criterios de salida.
- ✓ Iniciación de las acciones correctivas.
- ✓ Realizar decisiones.

1.6.4 Análisis y diseño de prueba

El análisis y diseño de prueba es la actividad donde los objetivos generales de prueba son transformados en condiciones de prueba y diseños de prueba tangibles.

El análisis y diseño de prueba tiene las siguientes tareas principales siguientes:

- ✓ Revisar la base de prueba (tales como requisitos, arquitectura, diseño, interfaces).
- ✓ Posibilidad de probar evaluación de los objetos de base de prueba y de prueba.
- ✓ Identificar las condiciones de prueba o los requisitos de prueba y los datos de prueba requeridos basado en el análisis de los elementos de prueba, la especificación, comportamiento y estructura.
- ✓ Diseñar las pruebas.
- ✓ Evaluar testabilidad de los requisitos y del sistema.
- ✓ Diseñar la configuración del entorno de prueba e identificar cualquier infraestructura y herramientas requeridas.

1.6.5 Implementación y ejecución de prueba

La implementación y ejecución de prueba es la actividad donde las condiciones de prueba son transformadas en casos de prueba y testware y el entorno es configurado.

La implementación y ejecución de prueba tienen las siguientes tareas principales:

- ✓ Desarrollar y priorizar casos de prueba, crear datos de prueba, escribir procedimientos de prueba y, opcionalmente, preparar armaduras de pruebas y escribir scripts de prueba automatizados.
- ✓ Crear conjuntos de prueba de los casos de prueba para la ejecución de prueba eficiente.
- ✓ Verificar que el entorno de prueba haya sido configurado correctamente.
- ✓ Ejecutar los casos de prueba ya sea manualmente o mediante el uso de herramientas de ejecución de prueba, de acuerdo con la secuencia planeada.
- ✓ Registrar el resultado de la ejecución de prueba y grabar las identidades y versiones del software bajo prueba, herramientas de prueba y testware.
- ✓ Comparar los resultados reales con los resultados esperados.
- ✓ Informar discrepancias como incidentes y analizarlos para establecer su causa (por ejemplo, un defecto en el código, en datos de prueba especificados, en el documento de prueba o un error en la forma en que la prueba fue ejecutada).
- ✓ Repetir las actividades de prueba como resultado de la acción tomada para cada discrepancia. Por ejemplo, la re-ejecución de una prueba que falló previamente para confirmar un arreglo (pruebas de confirmación), ejecución de una prueba corregida y/o ejecución de pruebas para asegurarse que los defectos no han sido introducidos en áreas no cambiadas del software o que el arreglo del defecto no reveló otros defectos (pruebas de regresión).

1.6.6 Evaluación de criterios de salida y reporte

Evaluar los criterios de salida es la actividad donde la ejecución de prueba es evaluada contra los objetivos definidos. Esto debe hacerse para cada nivel de prueba.

Evaluar los criterios de prueba tiene las siguientes tareas principales:

- ✓ Comprobar los registros de prueba contra los criterios de salida especificados en la planificación de prueba.
- ✓ Evaluar si más pruebas son necesitadas o si los criterios de salida especificados deberían ser cambiados.
- ✓ Escribir un reporte de resumen de prueba para las partes interesadas.

1.6.7 Actividades de cierre de prueba

Las actividades de cierre de prueba recolectan datos de las actividades de prueba completadas para consolidar experiencia, testware, hechos y números. Por ejemplo, donde un sistema de software es lanzado, un proyecto de prueba es completado (o cancelado), un hito ha sido alcanzado o un lanzamiento de mantenimiento ha sido completado.

Las actividades de cierre de prueba incluyen las siguientes tareas principales:

- ✓ Comprobar cuales entregables planeados han sido entregados, el cierre de los informes de incidentes o el aumento de registros de cambio para cualquiera que permanezca abierto y la documentación de la aceptación del sistema.
- ✓ Finalizar y archivar el testware, el entorno de prueba y la infraestructura de prueba para reutilización posterior.
- ✓ Entrega del testware a la organización de mantenimiento.
- ✓ Analizar las lecciones aprendidas para futuros lanzamientos y proyectos y la mejora de la madurez de pruebas.

1.7 Cuanto Testing es necesario?

- ✓ El testing exhaustivo es imposible.
- ✓ Testear todas las combinaciones de entradas y precondiciones no es factible.
- ✓ Para enfocar el testing nos debemos basar en Riesgos y Prioridades.

1.8 La psicología de prueba

1.8.1 Términos

Prueba independiente.

1.8.2 Introducción

El modo de pensar a ser usado mientras se prueba y revisa es diferente al aquel usado mientras se analiza o desarrolla. Con el correcto modo de pensar los desarrolladores pueden probar su propio código, mas la separación de su responsabilidad hacia un probador es realizada

típicamente para ayudar a enfocar el esfuerzo y para proporcionar beneficios adicionales, tales como una visión independiente por recursos de prueba profesionales y entrenados. La prueba independiente pude ser llevada a cabo en cualquier nivel de prueba.

Un cierto grado de independencia (evitando la tendencia del autor) es a menudo más efectivo al encontrar defectos y fallas. La independencia no es, sin embargo, un reemplazo para la familiaridad y los desarrolladores pueden encontrar eficientemente muchos defectos en su propio código. Varios niveles de independencia pueden ser definidos:

- ✓ Pruebas diseñadas por la(s) persona(s) que escribió (escribieron) el software bajo prueba (bajo nivel de independencia).
- ✓ Pruebas diseñadas por otra(s) persona(s) (por ejemplo, del equipo de desarrollo).
- ✓ Pruebas diseñadas por una(s) persona(s) de un grupo organizacional diferente (por ejemplo, un grupo de prueba independiente.
- ✓ Pruebas diseñadas por una(s) persona(s) de una organización o compañía diferente (es decir subcontratación o certificación por un organismo externo).

La gente y los proyectos son conducidos por objetivos. La gente tiende a alinear sus planes con los objetivos establecidos por la gestión y otras partes interesadas, por ejemplo, para encontrar defectos o para confirmar que el software trabaja. Por lo tanto, es importa definir claramente los objetivos de la prueba.

Identificar fallas durante la prueba puede ser percibido como criticismo contra el producto y contra el autor. La prueba es, por lo tanto, a menudo vista como una actividad destructiva, aunque es muy constructiva en la gestión de riesgos del producto. Buscar fallas en un sistema requiere curiosidad, pesimismo profesional, un ojo crítico, atención al detalle, buena comunicación con los pares de desarrollo y experiencia en que basar la conjetura de error.

Si los errores, defectos o fallas son comunicados en una forma constructiva, malos sentimientos entre los probadores y los analistas, diseñadores y desarrolladores pueden ser evitados. Esto se aplica a la revisión así como en la prueba.

El probador y el líder de prueba necesitan buenas habilidades interpersonales para comunicar información factual sobre los defectos, el progreso y los riesgos, en una forma constructiva. Para el autor del software o del documento, la información del defecto puede ayudarlo a mejorar sus habilidades. Los defectos encontrados y arreglados durante la prueba ahorrarán tiempo y dinero más tarde y reducirán riesgos.

Los problemas de comunicación pueden ocurrir, particularmente si los probadores son vistos como mensajeros de noticias no deseadas sobre los defectos. Sin embargo, existen varias formas para mejorar la comunicación y las relaciones entre los probadores y los demás:

- ✓ Iniciar con la colaboración en lugar de batallas – recordar a todos de la meta común de mejores sistemas de calidad.

- ✓ Comunicar los descubrimientos sobre el producto en una forma neutral enfocada en hechos sin criticar a la persona que lo creo, por ejemplo, escribir informes de incidentes factuales y objetivos y revisar los descubrimientos.
- ✓ Intentar de comprender como se siente la otra persona y porque reacciona como lo hace.
- ✓ Confirmar que la otra persona ha entendido que es lo que usted ha dicho y viceversa.

Referencias

1.1.5 Black, 2001, Kaner, 2002
1.2 Beizer, 1990, Black, 2001, Myers, 1979
1.3 Beizer, 1990, Hetzel, 1998, Myers, 1979
1.4 Hetzel, 1998
1.4.5 Black, 2001, Craig, 2002
1.5 Black, 2001, Hetzel, 1998