Probador Certificado

Programa de estudio de nivel básico

Versión 2010

International Software Testing Qualifications Board

Historial de Revisiones

Etiquetado	Fecha	Observaciones
ISTQB 2010	30-Mar-2010 fecha efectiva	Programa de estudio de nivel básico para probadores certificados – Versión de mantenimiento – Ver Apéndice E – Notas de versión del Programa de estudio 2010
ISTQB 2007	01-May-2007	Programa de estudio de nivel básico para probadores certificados – Versión de mantenimiento – Ver Apéndice E – Notas de versión del Programa de estudio 2007
ISTQB 2005	01-Jul-2005	Programa de estudio de nivel básico para probadores certificados
ASQF V2.2	Julio-2003	Programa de estudio ASQF de nivel básico. Versión 2.2 “Lehrplan Grundlagen des Software-testens“
ISEB V2.0	25-Feb-1999	Programa de estudio básico de pruebas de software ISEB. V2.025 febrero 1999

Índice general

Agradecimientos

El grupo de trabajo de Nivel Básico del Comité Internacional de Cualificación de Pruebas de Software (Edición 2010) agradece a: Thomas Müller (Presidente), Rahul Verma, Martin Klonk y Armin Beer; al equipo de revisión (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Tuula Pääkkšnen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veendendaal) y a todos los Comités nacionales por sus sugerencias a la versión actual del programa de estudio.

El grupo de trabajo de Nivel Básico del Comité Internacional de Cualificación de Pruebas de Software (Edición 2007) agradece a: Thomas Müller (Presidente), Dorothy Graham, Debra Friedenberg, y Erik van Veendendaal y el equipo de revisión (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson, y Wonil Kwon) y a todos los Comités nacionales por sus sugerencias.

El grupo de trabajo de Nivel Básico del Comité Internacional de Cualificación de Pruebas de Software (Edición 2005) agradece a: Thomas Müller (Presidente), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson y Erik van Veendendaal y al equipo de revisión y a todos los Comités nacionales por sus sugerencias.

Introducción a este Programa de estudio

Objetivo de este documento

El presente programa de estudio constituye la base para la cualificación internacional de pruebas de software en su nivel básico. El International Software Testing Qualifications Board (Comité Internacional de Cualificación de Pruebas de Software - ISTQB) recuerda a los comités nacionales la necesidad de acreditar a los formadores y formular las preguntas de los exámenes en su idioma local. Los formadores determinarán los métodos de enseñanza adecuados y presentarán el material didáctico para su acreditación. El programa de estudio ayudará a los candidatos a prepararse para el examen. Para más información sobre la historia y los antecedentes del programa de estudio, remítase al Apéndice A.

Probador certificado de nivel básico de pruebas de software

El título de nivel básico está dirigido a todas las personas que participan en el proceso de pruebas de software. Incluidas las personas que hacen las veces de probadores, analistas de pruebas, ingenieros de pruebas, consultores de pruebas, jefes de pruebas, probadores de aceptación de usuario y desarrolladores de software. Este título de nivel básico también es adecuado para todo aquel que desee obtener un entendimiento básico del proceso de pruebas de software, tales como jefes de proyecto, responsables de calidad, jefes de desarrollo de software, analistas de negocios, directores de TI y consultores de gestión. Los poseedores del Certificado de Nivel Básico podrán pasar a un nivel más alto de cualificación en el proceso de pruebas de software.

Objetivos de aprendizaje/Nivel de conocimiento cognitivo

En este programa de estudio se indican los objetivos de aprendizaje para cada sección. Éstos pueden clasificarse como sigue:

- K1: recordar, reconocer, retener
- K2: entender, explicar, razonar, comparar, clasificar, categorizar, dar ejemplos, resumir
- K3: aplicar, utilizar
- K4: analizar

Para más detalles y ejemplos sobre los objetivos de aprendizaje, remítase al Anexo B.

Todos los términos incluidos en la lista "Términos" debajo del título de cada capítulo deben ser recordados (K1), a pesar de que no aparezcan explícitamente mencionados en los objetivos de aprendizaje.

El examen

El examen del Certificado de Nivel Básico tomará como referencia este programa de estudio. Las respuestas a las preguntas del examen pueden exigir el uso de materiales basados en una o más secciones de este programa de estudio. Todas las secciones de este programa de estudio están sujetas a examen.

El examen será tipo test.

Los exámenes podrán realizarse como parte de un curso de formación acreditado o de manera independiente (por ejemplo, en un centro de exámenes o en un examen público). La compleción de un curso de formación acreditado no constituye un requisito previo para realizar el examen.

Acreditación

Un Comité nacional de ISTQB podrá acreditar a aquellos formadores cuyo material didáctico se ajuste al presente programa de estudio. Los formadores deberán atenerse a las directrices de acreditación estipuladas por el comité o por cualquier organismo que otorgue la acreditación. Por la presente se reconoce que los cursos acreditados se ajustan a este programa de estudio, por lo que podrán incluir un examen ISTQB.

Para más directrices para los formadores, remítase al Apéndice D.

Nivel de detalle

El nivel de detalle de este programa de estudio permite obtener un aprendizaje y unos exámenes consistentes a escala internacional. Con vistas a lograr este objetivo, el programa de estudio consta de:

- Objetivos formativos generales que describen la intención del Nivel Básico.
- Una lista de la información a enseñar, incluyendo una descripción y referencias a fuentes adicionales en caso necesario.
- Objetivos de aprendizaje de cada área de conocimiento que describan el resultado logrado en cuanto a aprendizaje cognitivo y a actitud.
- Una lista de los términos que los alumnos deben ser capaces de retener y entender.
- Una descripción de los conceptos clave a enseñar, incluyendo fuentes tales como literatura aceptada o normas.

El contenido del programa de estudio no constituye una descripción de toda el área de conocimiento del proceso de pruebas de software, sino que refleja el nivel de detalle a cubrir en los cursos de formación de nivel básico.

Organización del programa de estudio

Hay seis capítulos principales. En el título de cada capítulo se indica el máximo nivel de los objetivos de aprendizaje que se cubren en el capítulo y se especifica el tiempo que debe dedicarse al capítulo. Por ejemplo:

2. Pruebas durante todo el ciclo de vida del software (K2) 115 minutos

Este título quiere decir que el Capítulo 2 incluye objetivos de aprendizaje de K1 (asumidos cuando aparece un nivel superior) y K2 (pero no K3), y que el tiempo que debe dedicarse a enseñar el material incluido en el capítulo debe ser 115 minutos. Dentro de cada capítulo hay varias secciones. En cada sección también vienen indicados los objetivos de aprendizaje y la cantidad de tiempo requerida. Las subsecciones que no hacen referencia a la duración se consideran incluidas en el tiempo previsto para la sección.

1. Principios básicos del proceso de pruebas (K2)	155 minutos

Objetivos de aprendizaje de Principios básicos del proceso de pruebas

Los objetivos identifican lo que será usted capaz de hacer tras la compleción del módulo.

1.1 ¿Por qué es necesario el proceso de pruebas? (K2)

LO-1.1.1 Describir mediante ejemplos la forma en la que un defecto de software puede dañar a una persona, al medio ambiente o a una empresa (K2)

LO-1.10.2 Distinguir entre la causa raíz de un defecto y sus consecuencias (K2)

LO-1.10.3 Explicar mediante ejemplos por qué es necesario el proceso de pruebas (K2)

LO-1.10.4 Describir por qué el proceso de pruebas forma parte de los procedimientos de garantía de calidad y dar ejemplos de cómo las pruebas contribuyen a una mayor calidad (K2)

LO-1.10.5 Explicar y comparar mediante ejemplos los términos error, defecto, falta, fallo y los correspondientes términos error y bug (K2)

1.2 ¿En qué consiste el proceso de pruebas? (K2)

LO-1.2.1 Retener los objetivos comunes de las pruebas (K1)

LO-1.20.2 Poner ejemplos de los objetivos de las pruebas en las distintas fases del ciclo de vida del software (K2)

LO-1.20.3 Diferenciar las pruebas de depuración (K2)

1.3 Los siete principios del proceso de pruebas (K2)

LO-1.3.1 Explicar los siete principios del proceso de pruebas (K2)

1.4 Proceso básico de pruebas (K1)

LO-1.4.1 Retener las cinco actividades fundamentales y sus respectivas tareas desde la planificación hasta el cierre (K1)

1.5 La psicología del proceso de pruebas (K2)

LO-1.5.1 Retener los factores psicológicos que influyen en el éxito de las pruebas (K1)

LO-1.50.2 Contrastar la mentalidad de un probador y la de un desarrollador (K2)

1.1 ¿Por qué es necesario el proceso de pruebas? (K2)	*20 minutos*

Términos

Bug, defecto, error, fallo, falta, error, calidad, riesgo

1.1.1 Contexto de los sistemas de software (K1)

Los sistemas de software forman parte integrante de nuestras vidas, desde las aplicaciones comerciales (como la banca) hasta los productos de consumo (como los coches). A casi todos nos ha pasado alguna vez que un software no ha funcionado según lo previsto. Un software que no funciona correctamente puede dar lugar a muchos problemas, incluyendo la pérdida de dinero, tiempo o renombre, daños personales o incluso la muerte.

1.1.2 Causas de los defectos de software (K2)

Una persona puede cometer un error (error) que a su vez puede producir un defecto (falta, bug) en el código de programa o en un documento. Si se ejecuta un defecto en el código, el sistema puede no hacer lo que debiera (o hacer algo que no debiera), lo que provocaría un fallo. Algunos defectos de software, sistemas o documentos pueden dar lugar a fallos, pero no todos los defectos lo hacen.

Los defectos suceden porque las personas somos falibles y porque existen factores como presión, códigos complejos, infraestructuras complejas, tecnologías cambiantes y/o muchas interacciones del sistema.

Los fallos también pueden venir provocados por condiciones medioambientales. Así por ejemplo, la radiación, el magnetismo, los campos electrónicos y la contaminación pueden provocar faltas en sistemas o afectar a la ejecución de un software al modificar las condiciones del hardware.

1.1.3 Función del proceso de pruebas en el desarrollo, mantenimiento y operaciones de software (K2)

El hecho de someter los sistemas y la documentación a pruebas rigurosas puede ayudar a reducir el riesgo de complicaciones durante las operaciones y contribuir a la calidad del sistema de software, siempre que los defectos detectados se corrijan antes de que el sistema se ponga a disposición para su uso operativo.

Las pruebas de software también pueden servir para cumplir requisitos contractuales o legales, o estándares específicos del sector.

1.1.4 Proceso de pruebas y calidad (K2)

Gracias a las pruebas, se puede medir la calidad de un software en términos de los defectos

detectados por lo que respecta a requisitos y características funcionales y no funcionales (tales como fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad). Para más información sobre las pruebas no funcionales, remítase al Capítulo 2. Para más información sobre las características de software, remítase a "Ingeniería de software – Calidad de productos de software" (ISO 9126).

Las pruebas de software pueden aportar fiabilidad a la calidad del software en caso de detectar pocos o ningún defecto. Pasar una prueba diseñada correctamente reduce el nivel general de riesgo de un sistema. Sin embargo, si las pruebas detectan defectos, su corrección incrementará la calidad del sistema de software.

Debemos aprender de la experiencia de proyectos previos. Al entender las causas raíz de los defectos detectados en otros proyectos, pueden mejorarse procesos, lo que a su vez debería evitar que se repitieran dichos defectos y, en consecuencia, mejoraría la calidad de sistemas futuros. Se trata de un aspecto de la garantía de calidad.

Las pruebas deben estar integradas como una actividad más del proceso de garantía de calidad (es decir, deben estar al mismo nivel que el desarrollo de estándares, formación y análisis de defectos).

1.1.5 ¿Con cuántas pruebas es suficiente? (K2)

A la hora de determinar cuántas pruebas deben realizarse hay que tener en cuenta tanto los niveles de riesgo, incluyendo los riesgos técnicos, de seguridad y comerciales, como las limitaciones del proyecto, tales como el tiempo y el presupuesto. El tema del riesgo se trata más en profundidad en el Capítulo 5.

Las pruebas deben facilitar información suficiente a las partes interesadas para que éstas puedan adoptar decisiones informadas sobre el lanzamiento de un software o de un sistema probado, para pasar a la siguiente fase de desarrollo o para su entrega a los clientes.

1.2 ¿En qué consiste el proceso de pruebas? (K2)	*30 minutos*

Términos

Depuración, requisito, revisión, caso de prueba, pruebas, objetivo de prueba

Antecedentes

Por lo general se tiende a pensar que el proceso de pruebas consiste exclusivamente en ejecutar pruebas, es decir, ejecutar el software. Por supuesto, la ejecución de pruebas forma parte del proceso de pruebas, pero no representa la totalidad de las actividades incluidas en él.

Las actividades de pruebas se dan antes y después de la ejecución de la prueba. Entre estas actividades se encuentran: planificar y controlar, seleccionar las condiciones de las pruebas, diseñar y ejecutar casos de prueba, comprobar los resultados, evaluar los criterios de salida, elaborar informes sobre el proceso de pruebas y sobre el sistema probado, y finalizar o completar actividades de cierre una vez finalizada una fase de prueba. El proceso de pruebas también incluye la revisión de documentos (incluyendo el código fuente) y la realización de análisis estáticos.

También se puede recurrir a la realización de pruebas dinámicas y estáticas para lograr objetivos similares, las cuales generarán información que podrá utilizarse para mejorar tanto el sistema probado como el desarrollo y los procesos de prueba.

El proceso de pruebas puede tener los siguientes objetivos:

- Identificar defectos
- Aumentar la confianza en el nivel de calidad
- Facilitar información para la toma de decisiones
- Evitar la aparición de defectos

El proceso de reflexión y las actividades implicadas en el diseño de las pruebas al inicio del ciclo de vida del software (comprobando la base de prueba a través del diseño de pruebas) puede ayudar a evitar la introducción de defectos en el código. La revisión de documentos (por ejemplo, los requisitos) y la identificación y subsanación de problemas también pueden ayudar a evitar la aparición de defectos en el código.

Los distintos puntos de vista asociados al proceso de pruebas tienen en cuenta varios objetivos. Así por ejemplo, en las pruebas de desarrollo (por ejemplo, pruebas de componente, integración y sistemas), el principal objetivo puede ser provocar el máximo número de fallos al objeto de identificar y subsanar los defectos del software. En las pruebas de aceptación, el principal objetivo puede ser confirmar que el sistema funciona según lo

esperado o corroborar que se ajusta a los requisitos previstos. En algunos casos, el principal objetivo de las pruebas puede ser evaluar la calidad del software (sin intención alguna de arreglar posibles defectos) o facilitar información a las partes interesadas sobre el riesgo de lanzar el sistema en un momento dado. Las pruebas de mantenimiento a menudo incluyen pruebas para verificar que no han surgido nuevos defectos durante el desarrollo de los cambios. En las pruebas operativas, el principal objetivo puede ser evaluar características del sistema, tales como la fiabilidad o la disponibilidad.

Hay diferencia entre el proceso de depuración y el proceso de pruebas. Las pruebas dinámicas pueden identificar aquellos fallos que han sido provocados por defectos. La depuración es la actividad de desarrollo que localiza, analiza y elimina la causa del fallo. La posterior repetición de las pruebas por parte de un probador garantiza que la corrección llevada a cabo realmente elimina el fallo. La responsabilidad de cada una de estas actividades corresponde generalmente a los probadores, en el caso de las pruebas, y a los desarrolladores, en el caso de la depuración.

Para más detalles sobre el proceso de pruebas y sus actividades, remítase a la sección 1.4.

1.3 Siete principios para las pruebas (K2) — *35 minutos*

Términos

Pruebas exhaustivas

Principios

En los últimos 40 años se han propuesto una serie de principios que establecen pautas generales comunes a todas las pruebas.

Principio 1 – Las pruebas demuestran la presencia de defectos

Las pruebas pueden demostrar la que hay defectos, pero no pueden probar que no los hay. Las pruebas reducen la probabilidad de que haya defectos ocultos en el software pero, aunque no se detecte ningún defecto, no constituyen una evidencia de corrección.

Principio 2 – Las pruebas exhaustivas no existen

Probar todo (todas las combinaciones de entradas y precondiciones) es imposible, salvo en casos triviales. En lugar de pretender hacer pruebas exhaustivas, se deben realizar análisis de riesgos y prioridades para centralizar los esfuerzos de las pruebas.

Principio 3 – Pruebas tempranas

Para identificar los defectos en una etapa temprana, las actividades de pruebas se iniciarán lo antes posible en el ciclo de vida del software o del desarrollo del sistema, debiendo centrarse en objetivos definidos.

Principio 4 – Agrupación de defectos

Las pruebas deben concentrarse de manera proporcional en la densidad esperada, y más tarde observada, de los defectos de los módulos. Normalmente la mayor parte de los defectos detectados durante las pruebas previas al lanzamiento y la mayoría de los fallos operativos se concentran en un número reducido de módulos.

Principio 5 – Paradoja del pesticida

Si repetimos las mismas pruebas una y otra vez, eventualmente la misma serie de casos de prueba dejará de encontrar defectos nuevos. Para superar esta "paradoja del pesticida", los casos de prueba deben revisarse periódicamente y deben escribirse pruebas nuevas y diferentes para ejercitar distintas partes del software o del sistema con vistas a poder detectar más defectos.

Principio 6 – Las pruebas dependen del contexto

Las pruebas se llevan a cabo de manera diferente según el contexto. Así por ejemplo, la forma de probar un software crítico para la seguridad variará de la de un sitio de comercio electrónico.

Principio 7 – Falacia de ausencia de errores

La detección y corrección de defectos no servirá de nada si el sistema construido no es usable y no cumple las expectativas y necesidades de los usuarios.

1.4 Proceso básico de pruebas (K1)	*35 minutos*

Términos

Pruebas de confirmación, repetición de pruebas, criterios de salida, incidencia, pruebas de regresión, base de prueba, condición de prueba, cobertura de pruebas, datos de prueba, ejecución de pruebas, registro de pruebas, plan de pruebas, procedimiento de pruebas, política de pruebas, juego de prueba, informe resumen de pruebas, producto de soporte de prueba

Antecedentes

La parte más visible del proceso de pruebas es la ejecución de pruebas. No obstante, para ser efectivos y eficientes, los planes de pruebas también deben indicar el tiempo necesario para planificar las pruebas, diseñar los casos de prueba, preparar la ejecución y evaluar los resultados.

El proceso de pruebas básico consta de las siguientes actividades principales:

- Planificación y control
- Análisis y diseño
- Implementación y ejecución
- Evaluación de los criterios de salida e informes
- Actividades de cierre de pruebas

A pesar de tener una secuencia lógica, las actividades del proceso pueden solaparse o realizarse a la vez. Normalmente es necesario adaptar estas actividades al contexto del sistema y del proyecto.

1.4.1 Planificación y control de pruebas (K1)

La planificación de pruebas es la actividad de definir los objetivos de las pruebas y la especificación de las actividades de pruebas con vistas a cumplir los objetivos y la misión establecidos.

El control de pruebas es la actividad constante de comparar el progreso real con el plan previsto, e informar sobre el estado de las pruebas, incluyendo la existencia de desviaciones con respecto a lo que se había planificado. Implica la adopción de las medidas necesarias para cumplir la misión y los objetivos del proyecto. A efectos del control de pruebas, las actividades de pruebas deben monitorizarse a lo largo del proyecto. La planificación de pruebas tiene en cuenta el feedback de las actividades de control y seguimiento.

La planificación de pruebas y las tareas de control se definen en el Capítulo 5 de este programa de estudio.

1.4.2 Análisis y diseño de pruebas (K1)

El análisis y el diseño de pruebas es la activad durante la cual los objetivos de las pruebas generales se transforman en condiciones de prueba y casos de prueba tangibles.

La actividad de análisis y diseño de pruebas consta de las siguientes tareas principales:

- Revisar la base de pruebas 1 (tales como los requisitos, el nivel de integridad del software (nivel de riesgo), los informes de análisis de riesgos, la arquitectura, el diseño y las especificaciones de interfaz).
- Evaluar la testabilidad de la base de prueba y de los objetos de prueba.
- Identificar y priorizar las condiciones de prueba en base al análisis de los elementos de prueba, la especificación, el comportamiento y la estructura del software.
- Diseñar y priorizar los casos de prueba de alto nivel.
- Identificar los datos de prueba necesarios para soportar las condiciones de prueba y los casos de prueba.
- Diseñar la configuración del entorno de pruebas e identificar cualquier infraestructura y herramientas necesarias.
- Crear una trazabilidad bidireccional entre la base de pruebas y los casos de prueba.

1.4.3 Implementación y ejecución de pruebas (K1)

La implementación y ejecución de pruebas es la actividad en la que se especifican los procedimientos o guiones de prueba mediante la combinación de los casos de prueba en un orden determinado y la inclusión de cualquier otra información necesaria para la ejecución de las pruebas, se configura el entorno y se ejecutan las pruebas.

La implementación y ejecución de pruebas incluye las siguientes tareas principales:

- Finalizar, implementar y priorizar los casos de prueba (incluyendo la identificación de los datos de prueba).
- Desarrollar y priorizar procedimientos de prueba, crear datos de prueba y, de manera opcional, preparar arneses de pruebas y redactar guiones de prueba automatizados.
- Crear juegos de pruebas a partir de los procedimientos de prueba para lograr una ejecución de pruebas eficiente.
- Verificar que el entorno de pruebas ha sido correctamente configurado.

[1] El grado en que el software cumple o debe cumplir con una serie de características de software y/o características del sistema basadas en el software seleccionadas por las partes interesados (como por ejemplo, complejidad del software, evaluación de riesgos, nivel de seguridad, rendimiento deseado, fiabilidad o coste) que se definen para reflejar la importancia del software a las partes interesadas.

- Verificar y actualizar una trazabilidad bidireccional entre la base de pruebas y los casos de prueba.
- Ejecutar los procedimientos de prueba manualmente o recurriendo a herramientas de ejecución de pruebas, conforme a la secuencia prevista.
- Registrar los resultados de la ejecución de las pruebas y registrar las identidades y las versiones del software probado, así como las herramientas de prueba y los productos de soporte de pruebas.
- Comparar los resultados reales con los resultados esperados
- Reportar las discrepancias en forma de incidencias y analizarlas con vistas a establecer sus causas (por ejemplo, un defecto en el código o en los datos de pruebas especificados o en el documento de prueba, o un error en la forma en la que se ha ejecutado la prueba).
- Repetir las actividades de pruebas como resultado de una medida adoptada para cada discrepancia, por ejemplo, la repetición de una prueba que ha fallado anteriormente con vistas a confirmar que su corrección (pruebas de confirmación), la ejecución de una prueba corregida y/o la ejecución de pruebas con vistas a garantizar que los defectos no se han introducido en áreas no modificadas del software o que la subsanación del defecto no ha revelado la existencia de otros defectos (pruebas de regresión).

1.4.4 Evaluación de los criterios de salida e informes (K1)

La evaluación de los criterios de salida es la actividad que evalúa la ejecución de pruebas contra los objetivos definidos. Esta evaluación debería hacerse para cada nivel de prueba (remítase al apartado 2.2)

La evaluación de los criterios de salida consta de las siguientes tareas principales:

- Comprobar los registros de pruebas con los criterios de salida previstos en la planificación de la prueba.
- Evaluar si se requieren más pruebas o si deberían modificarse los criterios de salida especificados
- Elaborar un resumen de las pruebas para las partes interesadas.

1.4.5 Actividades de cierre de pruebas (K1)

Durante la fase de cierre de pruebas de un proceso se recopilan los datos de aquellas actividades de pruebas finalizadas con el objetivo de consolidar la experiencia, los productos de soporte de pruebas, los hechos y las cifras. Las actividades de cierre de pruebas tienen lugar en los hitos del proyecto, tales como el lanzamiento de un sistema de software, la finalización (o anulación) de un proyecto de pruebas, la consecución de un hito, o la finalización de una versión de mantenimiento.

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

- Comprobar cuáles de los productos entregables previstos han sido efectivamente entregados.
- Cerrar los informes de incidencias o aportar modificaciones a aquellos que siguen abiertos.
- Documentar la aceptación del sistema.
- Finalizar y archivar los productos de soporte de prueba, el entorno de pruebas y la infraestructura de pruebas para su posterior uso.
- Entregar los productos de soporte de prueba a la organización de mantenimiento.
- Analizar las lecciones aprendidas para determinar los cambios necesarios en futuras versiones y proyectos
- Utilizar la información recopilada para mejorar la madurez de las pruebas.

1.5 La psicología de las pruebas (K2)	*25 minutos*

Términos

Predicción de error, independencia

Antecedentes

La actitud que debe adoptarse durante la realización de las pruebas y la revisión difiere de la actitud que requiere el desarrollo del software. Si mantienen la actitud correcta, los desarrolladores pueden probar sus propios códigos. Sin embargo, normalmente esta responsabilidad se transfiere a un probador independiente con vistas a contribuir a centralizar los esfuerzos y obtener mayores beneficios, tales como una visión independiente de recursos formados y profesionales del ámbito de las pruebas. Las pruebas independientes pueden llevarse a cabo en cualquier nivel del proceso de pruebas.

Su determinado nivel de independencia (evitando el sesgo del autor) a menudo hace del probador la figura más efectiva a la hora de detectar defectos y fallos. No obstante, la independencia no constituye un sustituto del conocimiento, y los desarrolladores también pueden detectar muchos defectos en su propio código de manera eficiente. Pueden establecerse varios niveles de independencia según se muestra a continuación de menor a mayor:

- Pruebas diseñadas por las mismas personas que escribieron el software probado (bajo nivel de independencia).
- Pruebas diseñadas por terceros (por ejemplo, por miembros del equipo de desarrollo).
- Pruebas diseñadas por una persona procedente de otro grupo de organización (por

ejemplo, un equipo de pruebas independiente) o especialistas de pruebas (por ejemplo, especialistas en pruebas de usabilidad o rendimiento)

- Pruebas diseñadas por personas de otra organización o empresa (es decir, externalización o certificación por parte de un organismo externo)

Las personas y los proyectos se mueven por objetivos. Las personas tienden a ajustar sus planes a los objetivos establecidos por la dirección y demás personas interesadas, por ejemplo, para detectar errores o para confirmar que el software cumple sus objetivos. Por lo tanto, es importante establecer claramente cuáles son los objetivos de las pruebas.

Identificar fallos durante las pruebas puede percibirse como críticas contra el producto y contra el autor. En consecuencia, el proceso de pruebas a menudo se considera una actividad destructiva, a pesar de que es muy constructiva para la gestión de los riesgos del producto. La búsqueda de fallos en un sistema requiere curiosidad, pesimismo profesional, ojo crítico, atención al detalle, buena comunicación con los compañeros de desarrollo y

experiencia sobre la que basar la predicción de error.

Si los errores, defectos o fallos se comunican de una manera constructiva, puede evitarse la aparición de malentendidos entre los probadores y los analistas, los diseñadores y los desarrolladores. Esto es aplicable a los defectos detectados tanto durante las revisiones como durante las pruebas.

El probador y el líder de pruebas deben disponer de buenas aptitudes interpersonales para comunicar información objetiva sobre los defectos, el progreso y los riesgos de una manera constructiva. El autor del software o del documento puede aprovechar la información sobre defectos para mejorar sus propias habilidades. El hecho de localizar y corregir defectos durante el proceso de pruebas ahorrará tiempo y dinero más tarde, y reducirá riesgos.

Los problemas de comunicación pueden surgir, en particular, si los probadores se ven a sí mismos como meros mensajeros de malas noticias sobre defectos. No obstante, hay varias formas de mejorar la comunicación y las relaciones entre los probadores y los demás:

- Empezar juntos en lugar de enfrentados – recordar a todo el mundo el objetivo común de conseguir sistemas de mejor calidad.
- Comunicar los hallazgos del producto de una manera neutral y centrada en los hechos, sin criticar a las personas que lo crearon, por ejemplo, redactando informes de incidencias objetivos y revisando las conclusiones.
- Intentar comprender cómo se siente la otra persona y por qué reacciona de esa manera.
- Confirmar que la otra persona ha entendido lo que usted ha dicho y viceversa.

1.6 Código deontológico (K2)	*10 minutos*

El hecho de participar en el proceso de pruebas de software da acceso a información confidencial y privilegiada. El establecimiento de un código deontológico es necesario, entre otras cosas, para garantizar que la información no se utiliza de manera indebida. Reconociendo el código deontológico para ingenieros ACM e IEEE, el ISTQB® establece el siguiente código deontológico:

PÚBLICO – Los probadores de software certificados actuarán en coherencia con el interés público.

CLIENTE Y EMPLEADOR – Los probadores de software certificados actuarán en el mejor de los intereses de su cliente y empleador, en coherencia con el interés público.

PRODUCTO – Los probadores de software certificados garantizarán que los productos entregables que proporcionan (por lo que respecta a los productos y sistemas que prueban) se ajustan a los más altos estándares profesionales.

CRITERIO – Los probadores de software certificados mantendrán la integridad y la independencia en su criterio profesional.

GESTIÓN – Los jefes y líderes de pruebas de software certificados suscribirán y fomentarán un enfoque ético en la gestión de las pruebas de software.

PROFESIÓN – Los probadores de software certificados aumentarán la integridad y la reputación de la profesión en coherencia con el interés público.

COMPAÑEROS – Los probadores de software certificados serán equitativos y apoyarán a sus compañeros fomentando la cooperación con los desarrolladores de software.

NIVEL INDIVIDUAL – Los probadores de software certificados participarán en un aprendizaje a lo largo de toda la vida por lo que respecta a la práctica de su profesión y fomentarán la adopción de un enfoque ético en la práctica de su profesión.

Referencias

1.1.5 Black, 2001, Kaner, 2002

1.2 Beizer, 1990, Black, 2001, Myers, 1979

1.3 Beizer, 1990, Hetzel, 1988, Myers, 1979

1.4 Hetzel, 1988

1.4.5 Black, 2001, Craig, 2002

1.5 Black, 2001, Hetzel, 1988

2. Pruebas durante todo el ciclo de vida del software (K2)	115 minutos

Objetivos de aprendizaje de Pruebas durante todo el ciclo de vida del software

Los objetivos identifican lo que será usted capaz de hacer tras la compleción del módulo.

2.1 Modelos de desarrollo de software (K2)

LO-2.1.1 Explicar la relación existente entre desarrollo, actividades de pruebas y productos de trabajo en el ciclo de vida del desarrollo, poniendo ejemplos utilizando tipos de proyectos y productos (K2)

LO-2.10.2 Reconocer el hecho de que los modelos de desarrollo de software deben adaptarse al contexto de las características del proyecto y del producto (K1)

LO-2.10.3 Retener las características de buenas pruebas aplicables a cualquier modelo de ciclo de vida (K1)

2.2 Niveles de pruebas (K2)

LO-2.2.1 Comparar los distintos niveles de pruebas: Principales objetivos, objetos típicos de las pruebas, objetivos típicos de las pruebas (por ejemplo, funcionales o estructurales) y productos de trabajo asociados, personas que prueba, tipos de defectos y fallos a identificar (K2)

2.3 Tipos de pruebas (K2)

LO-2.3.1 Comparar cuatro tipos de pruebas de software (funcional, no funcional, estructural y asociado al cambio) a través de ejemplos (K2)

LO-2.3.2 Reconocer que las pruebas funcionales y estructurales se llevan a cabo en cualquier nivel de prueba (K1)

LO-2.3.3 Identificar y describir los tipos de pruebas no funcionales en base a requisitos no funcionales (K2)

LO-2.3.4 Identificar y describir los tipos de pruebas en base al análisis de la estructura o arquitectura de un sistema de software (K2)

LO-2.3.5 Describir el objetivo de las pruebas de confirmación y regresión (K2)

2.4 Pruebas de mantenimiento (K2)

LO-2.4.1 Comparar las pruebas de mantenimiento (pruebas a un sistema existente) con las pruebas de una aplicación nueva por lo que respecta a los tipos de pruebas, los desencadenantes de las pruebas y la cantidad de las pruebas (K2)

LO-2.4.2 Reconocer los indicadores de las pruebas de mantenimiento (modificación, migración y retirada) (K1)

LO-2.4.3. Describir la función de las pruebas de regresión y el análisis de impacto en el mantenimiento (K2).

2.1 Modelos de desarrollo de software (K2)	***20 minutos***

Términos

Software de distribución masiva (COTS), modelo de desarrollo iterativo-incremental, validación, verificación, modelo V

Antecedentes

El proceso de pruebas no es un proceso aislado; las actividades de pruebas están asociadas a actividades de desarrollo de software. El desarrollo de distintos modelos de ciclo de vida requiere distintos enfoques de pruebas.

2.1.1 Modelo V (Modelo de desarrollo secuencial) (K2)

A pesar de que existen variantes al modelo V, el tipo de modelo V generalizado se basa en cuatro niveles de pruebas correspondientes a los cuatro niveles de desarrollo.

Los cuatro niveles que se utilizan en este programa de estudio son los siguientes:

- Pruebas de componente (unidades)
- Pruebas de integración
- Pruebas de sistemas
- Pruebas de aceptación

En la práctica, un modelo V puede tener más, menos o diferentes niveles de desarrollo y pruebas, en función del proyecto y del producto de software. Así por ejemplo, pueden realizarse las pruebas de integración de componentes a continuación de las pruebas de componente, y las pruebas de integración de sistemas después de las pruebas de sistema.

Los productos de trabajo de software (tales como los escenarios empresariales o los casos de uso, las especificaciones requeridas, los documentos de diseño y los códigos) que se elaboran en la fase de desarrollo a menudo conforman la base de las pruebas en uno o más niveles de pruebas. Las referencias a productos de trabajo genéricos incluyen el Capability Maturity Model Integration (CMMI) o 'Procesos del ciclo de vida del software' (IEEE/IEC 12207). La verificación y validación (y el diseño de pruebas temprano) pueden realizarse durante el desarrollo de los productos de trabajo de software.

2.1.2 Modelos de desarrollo iterativo-incremental (K2)

El desarrollo iterativo-incremental es el proceso de establecer requisitos, diseñar, establecer y probar un sistema, realizado como una serie de ciclos de desarrollo más cortos. Algunos ejemplos son: Prototipos, Desarrollo Rápido de Aplicaciones (RAD), Proceso Unificado Racional (RUP) y modelos de desarrollo ágil. El sistema resultante producido por iteración puede ser probado en distintos niveles de prueba durante cada iteración. Un incremento, sumado a otros previamente desarrollados, constituye un sistema parcial creciente, que también debe ser probado. Después de la primera, las pruebas de regresión van adquiriendo importancia en todas las iteraciones. Los procesos de verificación y validación pueden llevarse a cabo para cada incremento.

2.1.3 Pruebas en un Modelo de Ciclo de Vida (K2)

En cualquier modelo de ciclo de vida se dan varias características de buenas pruebas:

- Para cada actividad de desarrollo existe una actividad de prueba correspondiente
- Cada nivel de prueba tiene objetivos de prueba específicos para dicho nivel
- Los procesos de análisis y diseño de las pruebas para un nivel de prueba dado deben iniciarse durante la actividad de desarrollo correspondiente.
- Los probadores deben iniciar su participación en la revisión de los documentos en cuanto haya borradores disponibles en el ciclo de vida de desarrollo.

Los niveles de prueba pueden combinarse o reorganizarse en función de la naturaleza del proyecto o de la arquitectura del sistema. Así por ejemplo, para la integración de un producto de software comercial de distribución masiva (COTS) en un sistema, el comprador puede realizar las pruebas de integración a nivel del sistema (por ejemplo, integración de la infraestructura y demás sistemas o despliegue del sistema) y las pruebas de aceptación (funcionales y/o no funcionales, y pruebas de usuario y/o operativas).

2.2 Niveles de prueba (K2)	*40 minutos*

Términos

Pruebas alfa, pruebas beta, pruebas de componente, controlador, pruebas de campo, requisito funcional, integración, pruebas de integración, requisito no funcional, pruebas de robustez, "stub", pruebas de sistema, entorno de pruebas, nivel de prueba, desarrollo guiado por pruebas, pruebas de aceptación de usuario.

Antecedentes

Por cada nivel de prueba, pueden identificarse los siguientes aspectos: los objetivos genéricos, los productos de trabajo a que se hace referencia para derivar los casos de prueba (es decir, la base de pruebas), el objeto de la prueba (es decir, lo que se está probando), los defectos y fallos típicos a detectar, los requisitos de arnés de pruebas y soporte de herramientas, y los enfoques específicos y responsabilidades.

En la fase de planificación de pruebas deberá tenerse en cuenta los datos de configuración del sistema, si dichos datos forman parte del sistema.

2.2.1 Pruebas de componente (K2)

Base de pruebas:

- Requisitos de componentes
- Diseño de detalle
- Código

Objetos de prueba típicos:

- Componentes
- Programas
- Conversión de datos / programas de migración

Módulos de bases de datos

Las pruebas de componente (también conocidas como pruebas de unidad, módulo o programa) tienen por objeto localizar defectos en y comprobar el funcionamiento de módulos de software, programas, objetos, clases, etc., que pueden probarse por separado. Pueden realizarse de manera independiente del resto del sistema, en función del contexto del ciclo de vida de desarrollo y del sistema. Para ello pueden utilizarse "stubs", controladores y simuladores.

Las pruebas de componente pueden incluir pruebas de funcionalidad y características no

funcionales específicas, tales como el comportamiento de recursos (por ejemplo, la búsqueda de filtraciones de memoria) o pruebas de robustez, además de pruebas estructurales (por ejemplo, cobertura de decisión). Los casos de prueba se derivan de productos de trabajo, tales como las especificaciones del componente, el diseño del software o el modelo de datos.

En general, las pruebas de componente se llevan a cabo mediante el acceso al código objeto de las pruebas y con el soporte de un entorno de desarrollo, como por ejemplo un marco de pruebas de unidad o una herramienta de depuración. En la práctica, las pruebas de componente generalmente cuentan con la participación del programador que escribió el código. En general, los defectos se corrigen en el momento en que se detectan, sin gestionarlos formalmente.

Un enfoque a seguir en las pruebas de componente es elaborar y automatizar los casos de prueba antes de codificarlos. Esto se denomina un primer enfoque de pruebas o un desarrollo guiado por pruebas. Este enfoque es altamente iterativo y está basado en la realización de ciclos de desarrollo de casos de prueba, para después construir e integrar pequeñas partes del código, y en la ejecución de las pruebas de componente corrigiendo cualquier problema e iterando hasta que se superen.

2.2.2 Pruebas de integración (K2)

Base de pruebas:

- Diseño de software y sistema
- Arquitectura
- Flujos de trabajo
- Casos de uso

Objetos de prueba típicos:

- Implementación de base de datos de subsistemas
- Infraestructura
- Interfaces

Configuración del sistema

- Datos de configuración

Las pruebas de integración se ocupan de probar las interfaces entre los componentes, las interacciones con distintas partes de un mismo sistema, como el sistema operativo, el sistema de archivos y el hardware, y las interfaces entre varios sistemas.

Puede haber más de un nivel de pruebas de integración y pueden llevarse a cabo en objetos de prueba de distinto tamaño, según se indica a continuación:

1. Las pruebas de integración de componentes se ocupan de probar las interacciones

entre los componentes del software y se realizan a continuación de las pruebas de componente.

2. Las pruebas de integración de sistema se ocupan de probar las interacciones entre los distintos sistemas o entre el hardware y el software, y pueden realizarse a continuación de las pruebas de sistema. En este caso, la organización de desarrollo puede controlar sólo una parte de la interfaz, lo que puede considerarse un riesgo. Los procesos de negocio implementados como flujos de trabajo pueden afectar a una serie de sistemas. Los problemas de plataforma transversales pueden ser importantes.

Cuanto más amplio sea el alcance de la integración, más difícil será asilar los fallos de un componente o sistema específico, lo que puede provocar un mayor riesgo y un tiempo adicional de diagnóstico.

Las estrategias de integración sistemáticas pueden basarse en la arquitectura de sistema (tales como de arriba hacia abajo y de abajo hacia arriba), tareas funcionales, secuencias de procesamiento de transacciones o cualquier otro aspecto del sistema o de los componentes. Con vistas a facilitar el aislamiento de faltas y realizar una detección temprana de los defectos, normalmente la integración será incremental en lugar de tipo “big-bang”.

Las pruebas de características específicas no funcionales (como el rendimiento) pueden incluirse tanto en las pruebas de integración como en las pruebas funcionales.

En cada fase de integración, los probadores deben concentrarse exclusivamente en la propia integración. Así por ejemplo, si están integrando el módulo A con el módulo B, deben concentrarse en probar la comunicación entre los módulos, no la funcionalidad del módulo individual, ya que eso ya se hizo durante las pruebas de componente. Para ello pueden utilizarse tanto el enfoque funcional como el enfoque estructural.

Idealmente, los probadores deben entender la arquitectura y modificar la planificación de la integración en consecuencia. Si las pruebas de integración se planifican antes de construir los componentes o sistemas, dichos componentes pueden construirse en el orden necesario para que las pruebas sean lo más eficientes posible.

2.2.3 Pruebas de sistema (K2)

Base de pruebas:

- Especificación de requisitos del sistema y software
- Casos de uso
- Especificaciones funcionales
- Informes de análisis de riesgos

Objetos de prueba típicos:

- Manuales de sistema, usuario y funcionamiento
- Configuración del sistema

Datos de configuración

Las pruebas de sistema se refieren al comportamiento de todo un sistema/producto. El alcance de las pruebas debe estar claramente indicado en el Plan Maestro de Pruebas y/o en el Plan de Pruebas de Nivel para cada nivel de prueba.

En las pruebas de sistema, el entorno de pruebas debe coincidir en la máxima medida posible con el objetivo final o con el último entorno de producción a fin de minimizar el riesgo de no identificar fallos específicos del entorno durante las pruebas.

Las pruebas de sistema pueden incluir pruebas basadas en riesgos y/o en especificaciones de requisitos, procesos de negocio, casos de uso u otras descripciones de texto de alto nivel o modelos de comportamiento de sistema, interacciones con el sistema operativo y recursos del sistema.

Las pruebas de sistema deben estudiar los requisitos funcionales y no funcionales del sistema y las características de calidad de los datos. Los probadores también deben enfrentarse a requisitos incompletos o no documentados. Las pruebas de sistema de los requisitos funcionales empiezan utilizando las técnicas basadas en la especificación (técnicas de caja negra) más apropiadas para el aspecto del sistema a probar. Así por ejemplo, puede crearse una tabla de decisión para las combinaciones de los efectos descritos en las normas de negocio. A continuación pueden utilizarse técnicas basadas en la estructura (técnicas de caja blanca) para evaluar la exhaustividad de las pruebas por lo que respecta a un elemento estructural, como por ejemplo una estructura de menú o la navegación de una página web (remítase al Capítulo 4).

A menudo las pruebas de sistema las realiza un equipo de pruebas independiente.

2.2.4 Pruebas de aceptación (K2)

Base de pruebas:

- Requisitos del usuario
- Requisitos del sistema
- Casos de uso
- Procesos de negocio
- Informes de análisis de riesgos

Objetos de prueba típicos:

- Procesos de negocio en sistema completamente integrado
- Procesos operativos y de mantenimiento
- Procedimientos de usuario
- Formularios
- Informes

Datos de configuración

Las pruebas de aceptación son a menudo responsabilidad de los clientes o usuarios de un sistema, a pesar de que también pueden participar otras partes interesadas.

El objetivo de las pruebas de aceptación es crear confianza en el sistema, partes del sistema o características específicas no funcionales del sistema. El objetivo principal de las pruebas de aceptación no es localizar defectos. Las pruebas de aceptación evalúan la buena disposición de un sistema para su despliegue y uso, a pesar de no constituir necesariamente el último nivel de prueba. Así por ejemplo, las pruebas de aceptación de un sistema pueden estar seguidas de una prueba de integración del sistema a gran escala.

Las pruebas de aceptación pueden darse en distintos momentos del ciclo de vida, tales como:

- Un producto de software COT puede ser objeto de pruebas de aceptación una vez instalado o integrado.
- Las pruebas de aceptación de la usabilidad de un componente pueden realizarse durante las pruebas de componente.
- Las pruebas de aceptación de una nueva mejora funcional pueden realizarse antes de las pruebas de sistema.

En general, las pruebas de aceptación pueden adoptar, entre otras, las siguientes formas:

Pruebas de aceptación de usuario

En general, verifican la idoneidad de uso del sistema por parte de los usuarios comerciales.

Pruebas operativas (de aceptación)

La aceptación del sistema por parte de los administradores del sistema, entre las que se incluyen:

- Pruebas de backup/restauración
- Recuperación de desastres
- Gestión de usuarios
- Tarea de mantenimiento
- Carga de datos y tareas de migración
- Comprobaciones periódicas de vulnerabilidades de seguridad

Pruebas de aceptación contractual y normativa

Las pruebas de aceptación contractual toman como base los criterios de aceptación previstos en un contrato para fabricar un software desarrollado a medida. Los criterios de aceptación deberán establecerse en el momento en que las partes aceptan contraer dicho contrato. Las pruebas de aceptación normativa toman como base cualquier normativa de obligado cumplimiento, tales como normativas gubernamentales, legales o de seguridad.

Pruebas alfa y beta (o de campo)

Los desarrolladores de software de mercado, o COTS, a menudo quieren obtener feedback

de clientes potenciales o existentes en su mercado antes de poner a la venta un producto de software. Las pruebas alfa se llevan a cabo en el emplazamiento de la organización de desarrollo, pero no las realiza el equipo de desarrollo. Las pruebas beta, o pruebas de campo, las realizan los clientes o clientes potenciales en sus propias instalaciones.

Las organizaciones también pueden emplear otros términos, tales como pruebas de aceptación en fábrica y pruebas de aceptación en emplazamiento, en el caso de sistemas probados antes y después de su traslado a las instalaciones del cliente.

2.3 Tipos de prueba (K2)	*40 minutos*

Términos

Pruebas de caja negra, cobertura de código, pruebas funcionales, pruebas de interoperabilidad, pruebas de carga, pruebas de mantenibilidad, pruebas de rendimiento, pruebas de portabilidad, pruebas de fiabilidad, pruebas de seguridad, pruebas de estrés, pruebas estructurales, pruebas de usabilidad, pruebas de caja blanca.

Antecedentes

Un grupo de actividades de pruebas pueden tener por objeto comprobar el sistema de software (o parte del sistema) en base a un motivo u objetivo específico.

Un tipo de prueba se centra en un objetivo de prueba en particular, que puede ser cualquiera de los siguientes:

- Una función a realizar por el software
- Una característica de calidad no funcional, tales como la fiabilidad o la usabilidad
- La estructura o arquitectura del software o sistema

Cambios asociados, es decir, confirmar que se han solucionado los defectos (pruebas de confirmación) y localizar cambios no intencionados (pruebas de regresión)

Puede desarrollarse y/o utilizarse un modelo del software en las pruebas estructurales (por ejemplo, un modelo de flujo de control o un modelo de estructura de menús), en las pruebas no funcionales (por ejemplo, un modelo de rendimiento, un modelo de amenaza de seguridad y un modelo de usabilidad), y en las pruebas funcionales (por ejemplo, un modelo de flujo de procesos, un modelo de transición de estados o una mera especificación de lenguaje).

2.3.1 Pruebas de funciones (Pruebas funcionales) (K2)

Las funciones que un sistema, subsistema o componente debe llevar a cabo pueden describirse en productos de trabajo tales como una especificación de requisitos, casos de uso o una especificación funcional, o incluso pueden no estar documentadas. Las funciones son “lo que” hace el sistema.

Las pruebas funcionales se basan en funciones y prestaciones (descritas en documentos o entendidas por los probadores) y en su interoperabilidad con sistemas específicos, y pueden llevarse a cabo en todos los niveles de prueba (por ejemplo, las pruebas de componente pueden basarse en una especificación de componente).

Las técnicas basadas en la especificación sirven para obtener condiciones de prueba y casos de prueba a partir de la funcionalidad de un software o sistema (remítase al Capítulo 4). Las

pruebas funcionales tienen en cuenta el comportamiento externo del software (pruebas de caja negra).

Un tipo de pruebas funcionales, las pruebas de seguridad, estudian las funciones (por ejemplo, un cortafuegos) asociadas a la detección de amenazas procedentes de personas ajenas y malintencionadas, tales como virus. Otro tipo de pruebas funcionales, las pruebas de interoperabilidad, evalúan la capacidad del producto de software de interactuar con uno o más componentes o sistemas especificados.

2.3.2 Pruebas de características no funcionales del software (Pruebas no funcionales) (K2)

Las pruebas no funcionales incluyen, pero sin limitarse a ello, pruebas de rendimiento, pruebas de carga, pruebas de estrés, pruebas de usabilidad, pruebas de mantenibilidad, pruebas de fiabilidad y pruebas de portabilidad. Estas pruebas se refieren a "cómo" funciona el sistema.

Las pruebas no funcionales pueden ejecutarse a todos los niveles de prueba. El término pruebas no funcionales hace referencia a las pruebas necesarias para medir las características de los sistemas y software que pueden cuantificarse según una escala variable, tales como los tiempos de respuesta en el caso de las pruebas de rendimiento. Estas pruebas pueden hacer referencia a un modelo de calidad como el previsto en "Ingeniería de software – Calidad de productos de software" (ISO 9126). Las pruebas no funcionales tienen en cuenta el comportamiento externo del software y, en la mayoría de los casos, para ello utiliza técnicas de diseño de pruebas de caja negra.

2.3.3 Pruebas de estructura/arquitectura de software (Pruebas estructurales) (K2)

Las pruebas estructurales (caja blanca) pueden realizarse en todos los niveles de prueba. Las técnicas estructurales son las más idóneas, después de las técnicas basadas en la especificación, para ayudar a medir la exhaustividad de las pruebas mediante una evaluación de la cobertura de un tipo de estructura.

La cobertura es la medida en que un juego de pruebas ha probado una estructura, expresada como porcentaje de los elementos cubiertos. Si la cobertura no es del 100%, entonces podrán diseñarse más pruebas para probar los elementos que faltan para aumentar la cobertura. Las técnicas de cobertura se abordan en el Capítulo 4.

En todos los niveles de prueba, pero especialmente en las pruebas de componente y en las pruebas de integración de componentes, puede recurrirse a herramientas para medir la cobertura de código de los elementos, tales como sentencias o decisiones. Las pruebas estructurales pueden basarse en la arquitectura del sistema, como por ejemplo una jerarquía de llamadas.

Los enfoques de las pruebas estructurales también pueden aplicarse a nivel de sistema, integración de sistemas o pruebas de aceptación.

2.3.4 Pruebas asociadas a cambios: Repetición de pruebas y pruebas de regresión (K2)

Una vez detectado y corregido un defecto, el software debe volver a probarse para confirmar que el defecto original ha sido corregido con éxito. A esto se le denomina confirmación. La depuración (corrección de defectos) es una actividad de desarrollo, no una actividad de pruebas.

Las pruebas de regresión son la prueba reiterada de un programa ya probado, después de haber sido modificado, con vistas a localizar defectos surgidos o no descubiertos como resultado del cambio o de los cambios. Estos defectos pueden estar en el software objeto de las pruebas, o en cualquier otro componente de software asociado o no asociado. Se realizan cuando el software, o su entorno, sufren modificaciones. El alcance de las pruebas de regresión depende del riesgo de no encontrar defectos en el software que antes funcionaba.

Las pruebas deben ser repetibles si desean utilizarse a efectos de las pruebas de confirmación o para dar soporte a las pruebas de regresión.

Las pruebas de regresión pueden realizarse en todos los niveles de prueba, e incluyen pruebas funcionales, no funcionales y estructurales. Los juegos de pruebas de regresión se ejecutan muchas veces y por lo general son de lenta evolución, por lo que las pruebas de regresión constituyen un gran potencial para la automatización.

2.4 Pruebas de mantenimiento (K2)	*15 minutos*

Términos

Análisis de impacto, pruebas de mantenimiento.

Antecedentes

Una vez desplegado, un sistema de software suele estar en servicio durante años o décadas. Durante todo este tiempo, el sistema, sus datos de configuración o su entorno son objeto de frecuentes modificaciones o ampliaciones. La planificación anticipada de versiones es crucial para el éxito de las pruebas de mantenimiento. Debe realizarse una distinción entre versiones planificadas y arreglos urgentes. Las pruebas de mantenimiento se realizan en un sistema operativo existente, y se activan a partir de modificaciones, casos de migración o la retirada del software o del sistema.

Entre las modificaciones se incluyen modificaciones de mejora planificadas (como por ejemplo, en base a las versiones), modificaciones correctivas y de emergencia y modificaciones de entorno, tales como actualizaciones previstas de sistema operativo previsto o bases de datos, actualizaciones previstas de software comercial de distribución masiva, o parches para corregir vulnerabilidades recientemente expuestas o descubiertas en el sistema operativo.

Las pruebas de mantenimiento en el caso de migración (por ejemplo de una plataforma a otra) deben incluir pruebas operativas tanto del nuevo entorno como del software modificado. Las pruebas de migración (pruebas de conversión) también son necesarias cuando se prevé migrar datos de otra aplicación al sistema objeto del mantenimiento.

Las pruebas de mantenimiento para la retirada de un sistema pueden incluir pruebas de migración de datos o archivo si se requieren largos períodos de retención de datos.

Además de probar lo que se ha modificado, las pruebas de mantenimiento incluyen pruebas de regresión ampliadas a partes del sistema que no han sido objeto de modificaciones. El alcance de las pruebas de mantenimiento variará en función del riesgo de la modificación, el tamaño del sistema existente y las dimensiones de la modificación. En función de las modificaciones, las pruebas de mantenimiento podrán realizarse en todos o en cualquier nivel de prueba y en todos o en cualquier tipo de prueba. La determinación de cómo el sistema existente podrá verse afectado por las modificaciones recibe el nombre de análisis de impacto y sirve para ayudar a decidir cuántas pruebas de regresión son necesarias. El análisis de impacto puede utilizarse para establecer el juego de pruebas de regresión.

Las pruebas de mantenimiento pueden resultar difíciles en caso de que no se disponga de especificaciones, o éstas estén desactualizados, o no haya probadores con conocimiento de dominio disponibles.

Referencias

2.1.3 CMMI, Craig, 2002, Hetzel, 1988, IEEE 12207

2.2 Hetzel, 1988

2.2.4 Copeland, 2004, Myers, 1979

2.3.1 Beizer, 1990, Black, 2001, Copeland, 2004

2.3.2 Black, 2001, ISO 9126

2.3.3 Beizer, 1990, Copeland, 2004, Hetzel, 1988

2.3.4 Hetzel, 1988, IEEE STD 829-1998

2.4 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE STD 829-1998

3. Técnicas estáticas (K2)	60 minutos

Objetivos de aprendizaje de Técnicas estáticas

Los objetivos identifican lo que será usted capaz de hacer tras la compleción del módulo.

3.1 Las técnicas estáticas y el proceso de pruebas (K2)

LO-3.1.1 Reconocer los productos de trabajo de software que pueden estudiarse en base a las distintas técnicas estáticas (K1)

LO-3.1.2 Describir la importancia y el valor de emplear técnicas estáticas para evaluar los productos de trabajo de software (K2)

LO-3.1.3 Explicar la diferencia entre técnicas estáticas y técnicas dinámicas, teniendo en cuenta los objetivos, los tipos de defectos a identificar y la función de estas técnicas en el marco del ciclo de vida del software (K2)

3.2 Proceso de revisión (K2)

LO-3.2.1 Retener las actividades, funciones y responsabilidades de una revisión formal estándar (K1)

LO-3.2.2 Explicar las diferencias entre los distintos tipos de revisión: revisión informal, revisión técnica, revisión guiada e inspección (K2)

LO-3.2.3 Explicar los factores para llevar a cabo revisiones con éxito (K2)

3.3 Análisis estático con herramientas (K2)

LO-3.3.1 Retener defectos y errores típicos identificados mediante el análisis estático y compararlos con los de las revisiones y pruebas dinámicas (K1)

LO-3.3.2 Describir, mediante ejemplos, los beneficios típicos del análisis estático (K2)

LO-3.3.3 Enumerar los defectos de código y diseño típicos que pueden identificarse mediante herramientas de análisis estático (K1)

3.1 Las técnicas estáticas y el proceso de pruebas (K2)	*15 minutos*

Términos

Pruebas dinámicas, pruebas estáticas

Antecedentes

Al contrario que las pruebas dinámicas, que exigen la ejecución de software, las técnicas de pruebas estáticas se basan en el examen manual (revisiones) y en el análisis automatizado (análisis estático) del código o de cualquier otra documentación del proyecto sin ejecutar el código.

Las revisiones constituyen una forma de probar los productos de trabajo del software (incluyendo el código) y pueden realizarse antes de ejecutar las pruebas dinámicas. Los defectos detectados durante las revisiones al principio del ciclo de vida (por ejemplo, los defectos encontrados en los requisitos) a menudo son mucho más baratos de eliminar que los detectados durante las pruebas realizadas ejecutando el código.

Una revisión podría hacerse íntegramente como una actividad manual, pero también existen herramientas de soporte. La principal actividad manual consiste en examinar un producto de trabajo y hacer comentarios al respecto. Cualquier producto de trabajo de software puede ser objeto de una revisión, incluyendo las especificaciones de requisitos, las especificaciones de diseño, el código, los planes de pruebas, las especificaciones de pruebas, los casos de pruebas, los guiones de prueba, las guías de usuario o las páginas web.

Los beneficios de las revisiones incluyen la detección y corrección temprana de defectos, el desarrollo de mejoras de productividad, la reducción de los tiempos de desarrollo, el ahorro de tiempo y dinero invertido en pruebas, el menor coste de la vida, menos defectos y comunicación mejorada. Las revisiones pueden encontrar omisiones, por ejemplo, en requisitos, que no suelen encontrarse en pruebas dinámicas.

Las revisiones, el análisis estático y las pruebas dinámicas tienen el mismo objetivo: identificar defectos. Se trata de procesos complementarios; las distintas técnicas pueden encontrar distintos tipos de defectos de una manera eficiente y efectiva. En comparación con las pruebas dinámicas, las técnicas estáticas localizan las causas o los fallos (defectos) más que los propios fallos.

Entre los defectos típicos que resultan más fáciles de localizar en revisiones que en las pruebas dinámicas se incluyen: desviaciones de los estándares, defectos de requisito, defectos de diseño, mantenibilidad insuficiente y especificaciones de interfaz incorrectas.

3.2 Proceso de revisión (K2)	*25 minutos*

Términos

Criterios de entrada, revisión formal, revisión informal, inspección, métrica, moderador, revisión entre pares, revisor, escriba, revisión técnica, revisión guiada

Antecedentes

Los distintos tipos de revisiones varían desde informales, que se caracterizan porque los revisores carecen de instrucciones escritas, hasta sistemáticas, que se caracterizan por incluir la participación del equipo, resultados documentados de la revisión y procedimientos documentados para llevar a cabo la revisión. La formalidad de un proceso de revisión está relacionada con factores tales como la madurez del proceso de desarrollo, cualquier requisito legal o normativo o la necesidad de un rastro de auditoría.

La forma en la que se lleva a cabo una revisión dependerá de los objetivos acordados de la revisión (por ejemplo, encontrar defectos, comprender, formar probadores y nuevos miembros del equipo, o debatir y decidir por consenso).

3.2.1 Actividades de una revisión formal (K1)

Una revisión formal típica incluye las siguientes actividades principales:

1. Planificar:
   - Definir los criterios de revisión
   - Seleccionar al personal
   - Asignar funciones
2. Definir los criterios de entrada y salida para más tipos de revisión formal (como por ejemplo, las inspecciones)
   - Seleccionar qué partes de los documentos deben revisarse
3. Inicio:
   - Repartir los documentos
   - Explicar los objetivos, procesos y documentos a los participantes
4. Comprobar los criterios de entrada (para tipos de revisión más formales)
5. Preparación individual
   - Prepararse para la reunión de revisión repasando los documentos

6. Prestar especial atención a posibles defectos, preguntas y comentarios
7. Examen/evaluación/registro de los resultados (reunión de revisión):
   - Debatir o registrar, mediante resultados documentados o actas (para tipos de revisión más formales)
   - Prestar especial atención a los defectos, hacer recomendaciones sobre cómo manejar los defectos, tomar decisiones
   - al respecto
8. Examinar/evaluar y registrar durante reuniones físicas o de seguimiento de los grupos comunicaciones electrónicas
9. Adaptar:
10. Corregir los defectos detectados (normalmente a cargo del autor)
    - Registrar el estado actualizado de los defectos (en revisiones formales)
11. Hacer un seguimiento:
    - Comprobar que los defectos han sido tratados
    - Recopilar métricas
12. Comprobar los criterios de salida (para tipos de revisión más formales)

3.2.2 Funciones y responsabilidades (K1)

Una revisión formal típica incluirá las siguientes funciones:

- Jefe: decide sobre la ejecución de las revisiones, asigna el tiempo en los calendarios del proyecto y determina si se han logrado los objetivos de la revisión.
- Moderador: es la persona que lidera la revisión del documento o serie de documentos, incluyendo la planificación de la revisión, la celebración de la reunión y el seguimiento después de la reunión. En caso necesario, el moderador podrá mediar entre los distintos puntos de vista y, con frecuencia, es la persona de la que depende el éxito de la revisión.
- Autor: es el escritor o la persona con máxima responsabilidad de los documentos a revisar.
- Revisores: se trata de personas con una experiencia específica en el ámbito técnico o comercial (también denominados evaluadores o inspectores) que, tras la necesaria preparación, identifican y describen las conclusiones (por ejemplo, los defectos) sobre el producto objeto de la revisión. Los revisores deben seleccionarse de manera que representen distintas perspectivas y funciones en el proceso de revisión, y deben participar en todas las reuniones de revisión.
- Escriba (o registrador): documenta todos los problemas, asuntos y puntos abiertos que se han identificado durante la reunión.

Abordar los productos de software o productos de trabajo asociados desde distintas perspectivas y utilizar listas de comprobación puede contribuir a la efectividad y eficiencia de las revisiones. Así por ejemplo, disponer de una lista de comprobación basada en distintas perspectivas como las del usuario, el mantenedor, el probador o las operaciones, o contar con una lista de comprobación que incluya los problemas típicos de requisitos puede ayudar a desvelar problemas no detectados anteriormente.

3.2.3 Tipos de revisiones (K2)

Un único producto de software o producto de trabajo asociado puede ser objeto de varias revisiones. Si se utiliza más de un tipo de revisión, el orden puede variar. Así por ejemplo, puede llevarse a cabo una revisión informal antes de la revisión técnica, o la inspección de una especificación de requisitos antes de una revisión guiada con el cliente. Las principales características, opciones y objetivos de los tipos de revisión disponibles son:

Revisión informal

- Ausencia de un proceso formal
- Puede adoptar la forma de programación por pares o una revisión por parte de un líder técnico de los diseños y el código
- Los resultados pueden estar documentados
- Su utilidad varía en función de los revisores
- Objetivo principal: forma barata de obtener beneficios

Revisión guiada

- Reunión liderada por el autor
- Puede adoptar la forma de escenarios, simulacros o participación de grupo de pares
- Sesiones abiertas
  - Preparación opcional de revisores previa a la reunión
  - Preparación opcional de un informe de revisión en el que se incluya la lista de conclusiones
- Escriba opcional (distinto del autor)
- Puede variar en la práctica desde bastante informal hasta muy formal
- Objetivos principales: aprender, entender, encontrar defectos

Revisión técnica

- Proceso documentado y definido para la detección de defectos que incluye la participación de pares y expertos técnicos, siendo la participación de la dirección opcional.
- Puede llevarse a cabo como una revisión entre pares sin la participación de la dirección
- Idealmente está dirigida por un moderador formado (distinto del autor)

- Preparación previa a la reunión por parte de los revisores
- Uso opcional de listas de comprobación
- Elaboración de un informe de revisión en el que se incluya la lista de conclusiones, el veredicto de si el producto de software cumple los requisitos y, si procede, recomendaciones en base a las conclusiones.
- Puede variar en la práctica desde bastante informal hasta muy formal
- Objetivos principales: debatir, tomar decisiones, evaluar alternativas, encontrar defectos, resolver problemas técnicos y comprobar la conformidad con las especificaciones, los planes, la normativa y los estándares.

Inspección

- Dirigida por un moderador formado (distinto del autor)
- Generalmente celebrada como un examen entre pares
- Funciones definidas
- Incluye una recopilación de métricas
- Proceso formal basado en normas y listas de comprobación
- Criterios de entrada y salida especificados para la aceptación del producto de software
- Preparación previa a la reunión
- Informe de inspección en el que se incluye una lista de las conclusiones
- Proceso de seguimiento formal
  - Proceso de mejora de componentes opcional
- Lector opcional
- Objetivo principal: identificar defectos

Las revisiones guiadas, las revisiones técnicas y las inspecciones pueden llevarse a cabo dentro de un mismo grupo de pares, es decir entre colegas de un mismo nivel organizativo. Este tipo de revisiones recibe el nombre de "revisión entre pares".

3.2.4 Factores de éxito de las revisiones (K2)

Entre los factores de éxito de las revisiones se encuentran los siguientes:

- Todas las revisiones tienen objetivos claros y predefinidos
- Se cuenta con la participación de las personas adecuadas para los objetivos de la revisión
- Los probadores constituyen revisores valiosos que contribuyen a la revisión y aprenden sobre el producto, lo que les permitirá preparar pruebas en una fase más temprana.
- Los defectos detectados son bienvenidos y se expresan de manera objetiva

- Se afrontan los problemas de personal y los aspectos psicológicos (por ejemplo, haciendo que sea una experiencia positiva para el autor)
- La revisión se lleva a cabo en una atmósfera de confianza ya que el resultado no se utilizará para evaluar a los participantes
- Se aplican las técnicas de revisión adecuadas para lograr los objetivos y para el tipo y nivel de productos de trabajo de software y los revisores
- Se utilizan listas de comprobación o funciones si procede para aumentar la efectividad del proceso de identificación de defectos
- Se ofrece formación sobre técnicas de revisión, especialmente por lo que respecta a las técnicas más formales como la inspección
- La dirección respalda la implantación de un buen proceso de revisión (por ejemplo, asignando el tiempo adecuado a las actividades de revisión en los calendarios del proyecto)
- Se hace hincapié en el aprendizaje y en la mejora del proceso

3.3 Análisis estático con herramientas (K2)	*20 minutos*

Términos

Compilador, complejidad, flujo de control, flujo de datos, análisis estático

Antecedentes

El objetivo del análisis estático es detectar defectos en el código fuente del software y en los modelos de software. El análisis estático se realiza sin que la herramienta llegue a ejecutar el software objeto de la revisión; en las pruebas dinámicas sí se ejecuta el código de software. El análisis estático permite identificar defectos difíciles de encontrar mediante pruebas dinámicas. Al igual que sucede con las revisiones, el análisis estático encuentra defectos en lugar de fallos. Las herramientas de análisis estático analizan el código del programa (por ejemplo, el flujo de control y flujo de datos) y las salidas generadas, tales como HTML o XML.

El valor del análisis estático es:

- La detección temprana de defectos antes de la ejecución de las pruebas
- Advertencia temprana sobre aspectos sospechosos del código o del diseño mediante el cálculo de métricas, tales como una medición de alta complejidad.
- Identificación de defectos que no se encuentran fácilmente mediante pruebas dinámicas
- Detectar dependencias e inconsistencias en modelos de software, como enlaces
- Mantenibilidad mejorada del código y del diseño
- Prevención de defectos, si se aprende la lección en la fase de desarrollo

Entre los defectos habitualmente detectados por las herramientas de análisis estático se encuentran:

- Referenciar una variable con un valor indefinido
- Interfaces inconsistentes entre módulos y componentes
- Variables que no se utilizan o que se declaran de forma incorrecta
- Código inaccesible (muerto)
- Ausencia de lógica o lógica errónea (posibles bucles infinitos)
- Construcciones demasiado complicadas
- Infracciones de los estándares de programación
- Vulnerabilidades de seguridad

- Infracciones de sintaxis del código y modelos de software

Las herramientas de análisis estático generalmente las utilizan los desarrolladores (cotejar con reglas predefinidas o estándares de programación) antes y durante las pruebas de componente y de integración, o durante la comprobación del código, para configurar las herramientas de gestión, así como los diseñadores durante la modelación del software. Las herramientas de análisis estático pueden producir un gran número de mensajes de advertencia que deben ser bien gestionados para permitir el uso más efectivo de la herramienta.

Los compiladores pueden constituir un soporte para los análisis estáticos, incluyendo el cálculo de métricas.

Referencias

3.2 IEEE 1028

3.2.2 Gilb, 1993, van Veenendaal, 2004

3.2.4Gilb, 1993, IEEE 1028

3.3 Van Veenendaal, 2004

4. Técnicas de diseño de pruebas (K4)	285 minutos

Objetivos de aprendizaje de Técnicas de diseño de pruebas

Los objetivos identifican lo que será usted capaz de hacer tras la compleción del módulo.

4.1 El proceso de desarrollo de pruebas (K3)

LO-4.1.1 Diferenciar entre una especificación de diseño de pruebas, una especificación de caso de prueba y una especificación de procedimiento de prueba (K2)

LO-4.1.2 Comparar los términos condición de prueba, caso de prueba y procedimiento de prueba (K2)

LO-4.1.3 Evaluar la calidad de los casos de prueba en términos de trazabilidad clara de los requisitos y resultados esperados (K2)

LO-4.1.4 Traducir casos de prueba en una especificación de procedimiento de prueba bien estructurada a un nivel de detalle relevante para el conocimiento de los probadores (K3)

4.2 Categorías de técnicas de diseño de pruebas (K2)

LO-4.2.1 Retener los motivos por los que resultan útiles tanto las técnicas de diseño de pruebas basadas en la especificación (caja negra) como las técnicas de diseño de pruebas basadas en la estructura (caja blanca) y enumerar las técnicas comunes para cada una de ellas (K1)

LO-4.2.2 Explicar las características, coincidencias y diferencias entre las pruebas basadas en la especificación, las pruebas basadas en la estructura y las pruebas basadas en la experiencia (K2)

4.3 Técnicas basadas en la especificación o técnicas de caja negra (K3)

LO-4.3.1 Escribir casos de prueba a partir de modelos de software dados aplicando partición de equivalencia, análisis de valores límites, tablas de decisión y diagramas/tablas de transición de estado (K3)

LO-4.3.2 Explicar el objetivo principal de cada una de las cuatro técnicas de pruebas, así como qué nivel y qué tipo de pruebas podría utilizar la técnica y cómo puede medirse la cobertura (K2)

LO-4.3.3 Explicar el concepto de las pruebas de caso de uso y sus ventajas (K2)

4.4 Técnicas basadas en la estructura o técnicas de caja blanca (K4)

LO-4.4.1 Describir el concepto y el valor de la cobertura de código (K2)

LO-4.4.2 Explicar los conceptos de sentencia y cobertura de decisión y explicar por qué

estos conceptos pueden utilizarse también en niveles de prueba que no sean pruebas de componente (por ejemplo, en procedimientos de negocio a nivel de sistema) (K2)

LO-4.4.3 Escribir casos de prueba a partir de flujos de control dados utilizando técnicas de diseño de pruebas de decisión y sentencia (K3)

LO-4.4.4 Evaluar la cobertura de sentencia y decisión para la integridad por lo que respecta a los criterios de salida definidos. K4)

4.5 Técnicas basadas en la experiencia (K2)

LO-4.5.1 Retener las causas para escribir casos de prueba en base a la intuición, la experiencia y el conocimiento de los defectos comunes (K1)

LO-4.5.2 Comparar técnicas basadas en la experiencia con técnicas de pruebas basadas en la especificación (K2)

4.6 Selección de la técnica de prueba (K2)

LO-4.6.1 Clasificar las técnicas de diseño de pruebas en función de su idoneidad en un contexto dado, para la base de prueba, los modelos respectivos y las características del software (K2)

4.1 El proceso de desarrollo de pruebas (K3)	*15 minutos*

Términos

Especificación de casos de prueba, diseño de pruebas, calendario de ejecución de pruebas, especificación de procedimiento de pruebas, guión de prueba, trazabilidad

Antecedentes

El proceso de desarrollo de pruebas que se describe en esta sección puede llevarse a cabo de varias maneras, desde muy informal, con poca o ninguna documentación, hasta muy formal (según se describe a continuación). El nivel de formalidad dependerá del contexto de las pruebas, incluyendo la madurez de las pruebas y de los procesos de desarrollo, las limitaciones de tiempo, los requisitos de seguridad o normativos y las personas implicadas.

Durante el análisis de pruebas, se analiza la documentación básica de las pruebas para determinar el objeto de las pruebas, es decir, para identificar las condiciones de prueba. Por condición de prueba se entiende un elemento o evento que debería ser verificado por uno o más casos de prueba (por ejemplo, una función, transacción, característica, atributo de calidad o elemento estructural).

Establecer la trazabilidad a partir de las condiciones de prueba en base a las especificaciones y a los requisitos permite tanto obtener un análisis de impacto efectivo cuando los requisitos cambian como determinar la cobertura de requisitos para una serie de pruebas. Durante el análisis de pruebas debe adoptarse un enfoque de pruebas detallado para seleccionar las técnicas de diseño de prueba a utilizar en función, entre otras consideraciones, de los riesgos identificados (remítase al Capítulo 5 para más información sobre análisis de riesgos).

Durante el diseño de pruebas, se crean y especifican los casos de prueba y los datos de prueba. Un caso de prueba consta de una serie de valores de entrada, precondiciones de ejecución, resultados esperados y condiciones de ejecución cuyo desarrollo tiene por objeto cubrir uno o varios objetivos de prueba o postcondiciones de prueba específicos. La "Norma para la documentación de prueba de software" (IEEE STD 829-1998) describe el contenido de las especificaciones de diseño de pruebas (incluyendo condiciones de prueba) y las especificaciones de los casos de prueba.

Los resultados esperados deben presentarse como parte de la especificación de un caso de prueba y deben incluir los valores de salida, las modificaciones a los datos y sentencias, además de cualquier otra consecuencia de la prueba. Si no se han definido los resultados esperados, se podrá interpretar como correcto un resultado plausible pero erróneo. Idealmente, los resultados esperados deben definirse antes de ejecutar la prueba.

Durante la ejecución de la prueba se desarrolla, implementan, priorizan y organizan los

casos de prueba en la especificación de procedimiento de prueba (IEEE STD 829-1998). El procedimiento de prueba especifica la secuencia de acciones necesarias para ejecutar una prueba. Si las pruebas se ejecutan utilizando una herramienta de ejecución de pruebas, la secuencia de acciones se especifica en un guión de prueba (que es un procedimiento de prueba automatizado).

Los distintos procedimientos de prueba y guiones de prueba automatizados conforman más tarde un calendario de ejecución de pruebas que establece el orden en el que deben ejecutarse los distintos procedimientos de prueba, y posiblemente también los guiones de prueba. El calendario de ejecución de pruebas tendrá en cuenta factores como las pruebas de regresión, la priorización y las dependencias técnicas y lógicas.

4.2 Categorías de técnicas de diseño de pruebas (K2)	15 minutos

Términos

Técnica de diseño de pruebas de caja negra, técnica de diseño de pruebas basadas en la experiencia, técnica de diseño de pruebas, técnica de diseño de pruebas de caja blanca

Antecedentes

El objetivo de una técnica de diseño de pruebas es identificar condiciones de prueba, casos de prueba y datos de prueba.

Tradicionalmente se distingue entre técnicas de pruebas de caja negra y de caja blanca. Las técnicas de diseño de prueba de caja negra (también conocidas como técnicas basadas en la especificación) son una forma de derivar y seleccionar condiciones de prueba, casos de prueba o datos de prueba en base a un análisis de la documentación básica de la prueba. Esto incluye pruebas tanto funcionales como no funcionales. Las pruebas de caja negra, por definición, no utilizan ninguna información sobre la estructura interna del componente o sistema a probar. Las técnicas de diseño de prueba de caja blanca (también conocidas como técnicas basadas en la estructura o estructurales) se basan en el análisis de la estructura del componente o sistema. Las pruebas de caja negra o de caja blanca también pueden basarse en la experiencia de los desarrolladores, probadores y usuarios para determinar el objeto de las pruebas.

Algunas técnicas pertenecen claramente a una única categoría, otras tienen elementos de varias categorías.

Este programa de estudio se refiere a las técnicas de diseño de pruebas basadas en la especificación como técnicas de caja negra y a las técnicas de diseño de pruebas basadas en la estructura como técnicas de caja negra. Además, se cubren las técnicas de diseño de pruebas basadas en la experiencia.

Algunas características comunes de las técnicas de diseño de pruebas basadas en la especificación son:

- El problema a solucionar, el software o sus componentes se especifican a partir de modelos, tanto formales como informales
- Los casos de prueba pueden obtenerse sistemáticamente a partir de estos modelos

Algunas características comunes de las técnicas de diseño de pruebas basadas en la estructura son:

- Información sobre cómo debe utilizarse el software construido para obtener los casos de prueba (por ejemplo, código e información de diseño detallada)
- Puede medirse el alcance de la cobertura del software para los casos de prueba

existentes, y puede obtenerse otros casos de prueba de manera sistemática para aumentar la cobertura.

Algunas características comunes de las técnicas de diseño de pruebas basadas en la experiencia son:

- Se emplea el conocimiento y la experiencia de las personas para obtener los casos de prueba
- El conocimiento de los probadores, desarrolladores, usuarios y demás partes interesadas sobre el software, su uso y su entorno constituye una fuente de información
- El conocimiento sobre los defectos probables y su distribución constituye otra fuente de información

4.3 Técnicas basadas en la especificación o técnicas de caja negra (K3)	***150 minutos***

Términos

Análisis de valores límites, pruebas de tabla de decisión, partición de equivalencia, pruebas de transición de estado, pruebas de caso de uso

4.3.1 Partición de equivalencia (K3)

En el caso de la partición de equivalencia, los valores de entrada del software o del sistema se dividen en grupos que se espera demuestren un comportamiento similar, de manera que puedan ser procesados de la misma forma. Las particiones de equivalencia (o clases) son aplicables a datos válidos, o valores que deben aceptarse, y datos no válidos, o valores que deben rechazarse. Las particiones también pueden aplicarse a los valores de salida, valores internos, valores relativos al tiempo (por ejemplo, antes o después de un evento) o a los parámetros de interfaz (por ejemplo, componentes integrados probados durante las pruebas de integración). Pueden diseñarse pruebas que cubran todas las particiones no válidas. La partición de equivalencia es aplicable a todos los niveles de pruebas.

La partición de equivalencia puede utilizarse para lograr objetivos de cobertura de entrada y salida. Puede utilizarse con entradas humanas, entradas vía interfaces a un sistema, o parámetros de interfaz en las pruebas de integración.

4.3.2 Análisis de valores límite (K3)

El comportamiento en el límite de cada partición de equivalencia tiene más posibilidades de ser incorrecto que el comportamiento dentro de la partición, por lo tanto los límites constituyen un área en el que las pruebas tenderán a incluir defectos. Los valores máximos y mínimos de una partición son sus valores límite. El valor límite de una partición válida

constituye un valor límite válido, mientras que los límites de una partición no válida constituyen valores límite no válidos. Las pruebas pueden diseñarse para cubrir valores límite tanto válidos como no válidos. Al diseñar casos de prueba, se selecciona una prueba por cada valor límite.

El análisis de valores límite puede aplicarse a todos los niveles de prueba. Es relativamente fácil de aplicar y su capacidad para detectar defectos es alta. Las especificaciones detalladas son útiles a la hora de establecer los límites interesantes.

Esta técnica a menudo se considera como una extensión de la partición de equivalencia u otras técnicas de diseño de pruebas de caja negra. Puede utilizarse en clases de equivalencia para los valores que los usuarios introducen en pantalla y, por ejemplo, en rangos de tiempo (tales como tiempo agotado o requisitos de velocidad de transacciones) o rangos de tabla (por ejemplo, el tamaño de la tabla es 256*256).

4.3.3 Pruebas de tabla de decisión (K3)

Las tablas de decisión constituyen una buena manera de reflejar los requisitos del sistema que contienen condiciones lógicas, y de documentar el diseño del sistema interno. Pueden utilizarse para registrar normas comerciales complejas que el sistema debe aplicar. Al crear las tablas de decisión, se analiza la especificación y se identifican las condiciones y acciones del sistema. Por lo general, las condiciones y acciones de entrada se sentencian de manera que deben ser verdaderas o falsas (Booleano). La tabla de decisión incluye las condiciones de activación, a menudo combinaciones de verdadero y falso para todas las condiciones de entrada, y las acciones resultantes para cada combinación de condiciones. Cada columna de la tabla corresponde a una regla comercial que define una combinación única de condiciones y que resulta en la ejecución de aquellas acciones asociadas a dicha regla. El estándar de cobertura que generalmente se utiliza en las tablas de decisión es disponer al menos de una prueba por columna en la tabla, lo que generalmente implica cubrir todas las combinaciones de condiciones de activación.

La ventaja de la tabla de decisión es que crea combinaciones de condiciones que de otro modo no se habrían practicado durante las pruebas. Puede aplicarse a todas las situaciones en aquellos casos en que la acción del software depende de varias decisiones lógicas.

4.3.4 Pruebas de transición de estado (K3)

Un sistema puede mostrar respuestas diferentes en función de sus condiciones actuales o su historial previo (su estado). En este caso, ese aspecto del sistema puede mostrarse con un diagrama de transición de estado. Esto permite al probador ver el software en términos de sus estados, transiciones entre estados, entradas o eventos que activan los cambios de estado (transiciones) y acciones que pueden derivarse de dichas transiciones. Los estados del sistema u objeto de la prueba son independientes, identificables y finitos en número.

Una tabla de estado muestra la relación entre los estados y las entradas, y eventualmente puede poner de manifiesto posibles transiciones no válidas.

Pueden diseñarse pruebas para cubrir una secuencia típica de estados, cubrir todos los estados, ejercitar todas las transiciones, ejercitar secuencias específicas de transiciones o

probar transiciones inválidas.

Las pruebas de transición de estado se utilizan mucho en la industria del software integrado y automatización técnica en general. Sin embargo, la técnica también resulta adecuada para modelar un objeto de negocio con estados específicos o para probar los flujos pantalla-diálogo (por ejemplo, en aplicaciones de Internet o escenarios comerciales).

4.3.5 Pruebas de caso de uso (K2)

Las pruebas pueden derivarse de casos de uso. Un caso de uso describe las interacciones entre los actores, incluyendo usuarios y el sistema, para producir un resultado de valor para un usuario de sistema o para el cliente. Los casos de uso pueden describirse tanto a nivel abstracto (caso de uso comercial, sin tecnología, nivel de proceso empresarial) como a nivel de sistema (caso de uso de sistema a nivel de funcionalidad del sistema). Cada caso de uso tiene precondiciones que deben cumplirse para que el caso de uso funcione correctamente. Cada caso de uso termina con postcondiciones, que son los resultados observables y el estado final del sistema una vez finalizado el caso de uso. Generalmente, un caso de uso tiene un escenario principal (o más probable) y caminos alternativos.

Los casos de uso describen los "flujos de proceso" mediante un sistema basado en su uso probable real, de manera que los casos de prueba derivados de los casos de uso resultan muy útiles a la hora de descubrir defectos en los flujos de proceso durante el uso real del sistema. Los casos de uso son de gran utilidad para diseñar las pruebas de aceptación con la participación del cliente/usuario. Asimismo, ayudan a descubrir eventuales defectos de integración, provocados por la interacción y la interferencia de distintos componentes, que las pruebas de componente a nivel individual no pueden detectar. El diseño de los casos de uso a partir de casos de uso puede combinarse con otras técnicas de pruebas basadas en la especificación.

4.4 Técnicas basadas en la estructura o técnicas de caja blanca (K4)	60 minutos

Términos

Cobertura de código, cobertura de decisión, cobertura de sentencia, pruebas basadas en la estructura

Antecedentes

Las pruebas basadas en la estructura o de caja blanca se basan en una estructura identificada del software o del sistema, según se demuestra en los siguientes ejemplos:

- Nivel de componente: la estructura de un componente de software, es decir, sentencias, decisiones, ramas o incluso caminos distintos
- Nivel de integración: la estructura puede ser un árbol de llamadas (un diagrama en el que los módulos llaman a otros módulos)
- Nivel de sistema: La estructura puede ser una estructura de menús, procesos de negocio o una estructura de página web

En este apartado, se tratan tres técnicas de diseño de pruebas estructurales relacionadas con códigos para la cobertura de código, en base a sentencias, ramas y decisiones. Para las pruebas de decisiones, puede utilizarse un diagrama de flujo para visualizar las alternativas para cada decisión.

4.4.1 Pruebas de sentencias y cobertura (K4)

En las pruebas de componente, la cobertura de sentencia es la evaluación del porcentaje de sentencias ejecutables que han sido practicadas por un juego de caso de prueba. La técnica de pruebas de sentencia obtiene casos de prueba para ejecutar sentencias específicas, normalmente con vistas a aumentar la cobertura de sentencias.

La cobertura de sentencia viene determinada por el número de sentencias ejecutables cubiertas (diseñadas o ejecutadas) por casos de prueba dividido entre el número de todas las sentencias ejecutables en el código objeto de la prueba.

4.4.2 Pruebas de decisión y cobertura (K4)

La cobertura de decisión, asociada a las pruebas de rama, es la evaluación del porcentaje de los resultados de decisión (por ejemplo, las opciones verdadero y falso de una sentencia SI) que han sido practicados por un juego de caso de prueba. La técnica de prueba de decisión obtiene los casos de prueba para ejecutar resultados de decisiones específicos. Las ramas forman puntos de decisión en el código.

La cobertura de decisión viene determinada por el número total de resultados de decisión

cubiertos (diseñados o ejecutados) por casos de prueba dividido entre el número total de posibles resultados de decisión en el código objeto de la prueba.

Las pruebas de decisión constituyen una forma de pruebas de flujo de control ya que siguen un flujo específico de control a través de los puntos de decisión. La cobertura de decisión es más fuerte que la cobertura de sentencia; un 100% de cobertura de decisión garantiza un 100% de cobertura de sentencia, pero no al contrario.

4.4.3 Otras técnicas basadas en la estructura (K1)

Existen niveles más fuertes de cobertura estructural además de la cobertura de decisión, como por ejemplo la cobertura de condición y la cobertura de condición múltiple.

El concepto de cobertura también puede aplicarse a otros niveles de prueba. Así por ejemplo, en el nivel de integración, el porcentaje de módulos, componentes o clases practicados por un juego de caso de prueba podría expresarse como módulo, componente o cobertura de clase.

El uso de herramientas constituye un soporte útil para las pruebas estructurales del código.

4.5 Técnicas basadas en la experiencia (K2)	*30 minutos*

Términos

Pruebas exploratorias, ataque (falta)

Antecedentes

Las pruebas basadas en la experiencia son aquellas en las que las pruebas se derivan de la habilidad e intuición del probador y de su experiencia con aplicaciones y tecnologías similares. Si se utilizan para aumentar las técnicas sistemáticas, estas técnicas pueden ser útiles a la hora de identificar pruebas especiales que no pueden capturarse fácilmente mediante técnicas formales, especialmente si se aplican después de adoptar enfoques más formales. No obstante, esta técnica puede tener distintos grados de efectividad, en función de la experiencia del probador.

Una técnica basada en la experiencia muy usada es la predicción de error. En general, los probadores anticipan los defectos en base a su experiencia. Un enfoque estructurado de la técnica de predicción de error consiste en enumerar una lista de posibles defectos y diseñar pruebas para atacar dichos defectos. Este enfoque sistemático se denomina ataque de faltas. Esta lista de defectos y fallos puede elaborarse en base a la experiencia, a los datos disponibles sobre defectos y fallos y a partir del conocimiento común sobre por qué falla el software.

Las pruebas exploratorias coinciden con las fases de diseño de pruebas, ejecución de pruebas, registro de pruebas y aprendizaje, en base a un contrato de pruebas que contempla los objetivos de las pruebas, y se llevan a cabo dentro de los períodos de tiempo establecidos. Se trata de un enfoque especialmente útil en los casos en los que las especificaciones son escasas o inadecuadas y existe una importante presión temporal, o para aumentar o complementar otras pruebas más formales. Asimismo, puede servir como comprobación del proceso de pruebas, para ayudar a garantizar que los defectos más graves han sido efectivamente detectados.

4.6 Selección de técnica de prueba (K2)	*15 minutos*

Términos

Ningún término específico.

Antecedentes

La selección de la técnica de pruebas a utilizar depende de una serie de factores entre los que se incluyen el tipo de sistema, los estándares normativos, los requisitos contractuales o de cliente, el nivel de riesgo, el tipo de riesgo, el objetivo de prueba, la documentación disponible, el conocimiento de los probadores, el tiempo y el presupuesto, el ciclo de vida de desarrollo, los modelos de caso de uso y la experiencia previa en los tipos de defectos detectados.

Algunas técnicas resultan más aplicables a ciertas situaciones y niveles de prueba, mientras que otras son aplicables a todos los niveles de pruebas.

Para crear casos de prueba, los probadores generalmente aplican una combinación de técnicas de pruebas, entre las que se incluyen técnicas guiadas por procesos, reglas y datos, con vistas a garantizar la correcta cobertura del objeto que se está probando.

Referencias

4.1 Craig, 2002, Hetzel, 1988, IEEE STD 829-1998

4.2 Beizer, 1990, Copeland, 2004

4.3.1 2.2.4 Copeland, 2004, Myers, 1979

4.3.2 Copeland, 2004, Myers, 1979

4.3.3 Beizer, 1990, Copeland, 2004

4.3.4 Beizer, 1990, Copeland, 2004

4.3.5 Copeland, 2004

4.4.3 Beizer, 1990, Copeland, 2004

4.5 Kaner, 2002

4.6 Beizer, 1990, Copeland, 2004

5. Gestión de pruebas (K3)	170 minutos

Objetivos de aprendizaje de Gestión de pruebas

Los objetivos identifican lo que será usted capaz de hacer tras la compleción del módulo.

5.1 Organización de pruebas (K2)

LO-5.1.1 Reconocer la importancia de las pruebas independientes (K1)

LO-5.1.2 Explicar las ventajas y desventajas de las pruebas independientes en el seno de una organización (K2)

LO-5.1.3 Reconocer los distintos miembros del equipo a tener en cuenta para crear un equipo de pruebas (K1)

LO-5.1.4 Retener las tareas del jefe de pruebas y del probador estándar (K1)

5.2 Planificación y estimación de pruebas (K3)

LO-5.2.1 Reconocer los distintos niveles y objetivos de la planificación de pruebas (K1)

LO-5.2.2 Resumir el objetivo y el contenido del plan de pruebas, la especificación de diseño de pruebas y los documentos de procedimiento de prueba de conformidad con la “Norma para la documentación de prueba de software" (IEEE Std 829-1998) (K2)

LO-5.2.3 Diferenciar entre enfoques de pruebas conceptualmente diferentes, tales como los analíticos, los basados en modelos, los metódicos, los que se ajustan a procesos/estándares, los dinámicos/heurísticos, los consultativos y los anti-regresión (K2)

LO-5.2.4 Diferenciar entre el objeto de la planificación de pruebas de un sistema y el establecimiento del calendario de ejecución de las pruebas (K2)

LO-5.2.5 Escribir un calendario de ejecución de las pruebas para una serie dada de casos de prueba, teniendo en cuenta la priorización y las dependencias técnicas y lógicas (K3)

LO-5.2.6 Enumerar las actividades de preparación y ejecución de pruebas que deben tenerse en cuenta durante la planificación de las pruebas (K1)

LO-5.2.7 Retener los factores típicos que influyen en el esfuerzo asociado a las pruebas (K1)

LO-5.2.8 Diferenciar entre dos enfoques de estimación conceptualmente diferentes. El enfoque basado en métricas y el enfoque basado en expertos (K2)

LO-5.2.9 Reconocer/justificar los criterios de entrada y salida adecuados para niveles de

prueba específicos y grupos de casos de prueba (por ejemplo, para pruebas de integración, pruebas de aceptación o casos de prueba para las pruebas de usabilidad) (K2)

5.3 Seguimiento y control del progreso de las pruebas (K2)

LO-5.3.1 Retener las métricas comúnmente utilizadas para hacer un seguimiento de la preparación y ejecución de pruebas (K1)

LO-5.3.2 Explicar y comparar las métricas de prueba para la elaboración de informes de prueba y el control de las pruebas (por ejemplo, los defectos encontrados y corregidos, y las pruebas superadas y no superadas) por lo que respecta al objetivo y al uso (K2)

LO-5.3.3 Resumir el objetivo y el contenido del informe resumen de pruebas de conformidad con la "Norma para la documentación de prueba de software" (IEEE Std 829-1998) (K2)

5.4 Gestión de la configuración (K2)

LO-5.4.1 Resumir cómo la gestión de la configuración da soporte a las pruebas (K2)

5.5 Riesgos y pruebas (K2)

LO-5.5.1 Describir un riesgo como un posible problema que amenazaría la consecución de uno o más objetivos de proyecto de las partes interesadas (K2)

LO-5.5.2 Recordar que el nivel de riesgo viene determinado por la probabilidad (de suceder) y el impacto (daño resultante si llega a suceder) (K1)

LO-5.5.3 Distinguir entre riesgos de proyecto y producto (K2)

LO-5.5.4 Reconocer los riesgos típicos de producto y proyecto (K1)

LO-5.5.5 Describir, mediante ejemplos, cómo puede utilizarse el análisis de riesgos y la gestión de riesgos para la planificación de pruebas (K2)

5.6 Gestión de incidencias (K3)

LO-5.6.1 Reconocer el contenido de un informe de incidencias de conformidad con la “Norma para la documentación de prueba de software" (IEEE Std 829-1998) (K1)

LO-5.6.2 Redactar un informe de incidencias sobre la observación de un fallo durante el proceso de pruebas (K3)i

5.1 Organización de pruebas (K2)	*30 minutos*

Términos

Probador, líder de pruebas, jefe de pruebas

5.1.1 Organización de pruebas e independencia (K2)

La efectividad de la identificación de defectos del proceso de pruebas y revisión puede mejorarse si se utilizan probadores independientes. Entre las alternativas para obtener independencia se incluyen las siguientes:

- Ausencia de probadores independientes: los desarrolladores son los que prueban su propio código
- Probadores independientes dentro de los equipos de desarrollo
- Equipos de prueba independientes o grupos dentro de la organización, que reporten a la dirección de proyectos o a la dirección ejecutiva
- Probadores independientes procedentes de la organización comercial o de la comunidad de usuarios
- Especialistas en pruebas independientes para tipos de pruebas específicas, tales como probadores de usabilidad, probadores de seguridad o probadores de certificación (que certifican un producto de software en base a los estándares y la normativa)
- Los probadores independientes subcontratados o externos a la organización

Para proyectos grandes, complejos o críticos para la seguridad, normalmente lo mejor es contar con varios niveles de pruebas y poner alguno o todos los niveles a cargo de probadores independientes. El personal de desarrollo puede participar en las pruebas, especialmente en los niveles más bajos, pero su falta de objetividad a menudo limita su efectividad. Los probadores independientes pueden tener potestad para exigir y definir procesos y reglas de prueba; no obstante, los probadores sólo deberán asumir dicha función en relación con los procesos previa orden de la dirección en ese sentido.

Algunas ventajas de la independencia son:

- Los probadores independientes ven más y diferentes defectos y son objetivos
- Un probador independiente puede comprobar los supuestos planteados durante las fases de especificación e implementación del sistema.

Algunos inconvenientes son:

- Aislamiento del equipo de desarrollo (si se trata como totalmente independiente)
- Los desarrolladores pueden llegar a perder el sentido de responsabilidad frente a la

calidad

- Los probadores pueden verse como cuellos de botella o ser culpados de retrasos en el lanzamiento.

Las tareas de prueba pueden realizarlas personas con una función de pruebas específica, o por otra persona con otro cargo como jefe de proyecto, jefe de calidad, desarrollador, experto de negocio y dominio, operaciones de infraestructura o TI.

5.1.2 Tareas del Líder de Pruebas y del Probador (K1)

Este programa de estudio cubre dos cargos de pruebas, el líder de pruebas y el probador. Las actividades y las tareas llevadas a cabo por las personas que ocupan estos dos cargos dependen del proyecto y del contexto del producto, las personas en los cargos y la organización.

A veces el líder de prueba recibe el nombre de jefe de pruebas o coordinador de pruebas. El cargo de líder de pruebas puede desempeñarse por un jefe de proyecto, un jefe de desarrollo, un jefe de garantía de calidad o el jefe de un grupo de pruebas. En proyectos grandes, puede haber dos cargos: líder de pruebas y jefe de pruebas. Normalmente, el líder de pruebas es el que planifica, monitoriza y controla las actividades y tareas de prueba según lo descrito en la Sección 1.4.

Las tareas típicas del líder de pruebas incluyen, entre otras:

- Coordinar la estrategia de pruebas y planificar con los jefes de proyecto y demás responsables
- Redactar o revisar una estrategia de pruebas para el proyecto, y una política de pruebas para la organización.
- Contribuir a la perspectiva de pruebas de otras actividades de proyecto, tales como la planificación de integración.
- Planificar las pruebas – teniendo en cuenta el contexto y entendiendo los objetivos y riesgos de las pruebas - incluyendo seleccionar los enfoques de prueba, calcular el tiempo, el esfuerzo y el coste de las pruebas, contratar recursos, definir niveles de prueba, ciclos y planificar la gestión de incidencias.
- Iniciar la especificación, preparación, implementación y ejecución de pruebas, hacer un seguimiento de los resultados de pruebas y comprobar los criterios de salida.
- Adaptar la planificación en base a los resultados y progreso de las pruebas (a veces documentados en informes de estado) y adoptar todas las acciones necesarias para compensar los problemas.
- Establecer una gestión de la configuración adecuada de los productos de soporte de pruebas a efectos de su trazabilidad.
- Introducir métricas adecuadas para medir el progreso de las pruebas y evaluar la calidad de las pruebas y del producto
- Decidir qué debe automatizarse, hasta qué punto y cómo

- Seleccionar las herramientas de soporte de las pruebas y organizar cursos de formación sobre el uso de dichas herramientas para los probadores.
- Decidir sobre la implementación del entorno de pruebas
- Redactar informes resúmenes de pruebas en base a la información recopilada durante las pruebas

Las tareas típicas del probador incluyen, entre otras:

- Revisar y contribuir a los planes de pruebas
- Analizar, revisar y evaluar los requisitos de usuario, las especificaciones y los modelos para su testabilidad.
- Crear especificaciones de prueba
- Configurar el entorno de pruebas (a menudo en coordinación con administración del sistema y gestión de redes)
- Preparar y obtener datos de prueba
- Implementar pruebas en todos los niveles de prueba, ejecutar y registrar las pruebas, evaluar los resultados y documentar las desviaciones de los resultados esperados
- Utilizar herramientas de administración o gestión y herramientas de seguimiento de prueba según proceda
- Automatizar pruebas (puede contar con el soporte de un desarrollador o un experto en automatización de pruebas)
- Medir el rendimiento de los componentes y sistemas (si procede)
- Revisar las pruebas desarrolladas por terceros

Las personas dedicadas al análisis de pruebas, al diseño de pruebas, a tipos de pruebas específicas o a la automatización de pruebas pueden ser especialistas en estas tareas. En función del nivel de prueba y de los riesgos asociados al producto y al proyecto, hay distintas personas que pueden asumir el papel de probador manteniendo cierto grado de independencia. En general, los probadores a nivel de componente e integración son los desarrolladores, los probadores a nivel de aceptación de pruebas son los expertos de negocio y usuarios, y los probadores de pruebas de aceptación operativas son los operadores.

5.2 Planificación y estimación de pruebas (K3)	*40 minutos*

Términos

Enfoque de pruebas, estrategia de pruebas

5.2.1 Planificación de pruebas (K2)

Este apartado describe el objetivo de la planificación de pruebas en el ámbito del desarrollo y la implementación de proyectos, así como en las actividades de mantenimiento. La planificación podrá documentarse en un plan de pruebas maestro y en planes de prueba por separado para niveles de pruebas tales como pruebas de sistema y pruebas de aceptación. El diseño de un documento de planificación de pruebas se aborda en la "Norma para la documentación de prueba de software" (IEEE Std 829- 1998).

La planificación se ve afectada por la política de pruebas de la organización, el alcance de las pruebas, los objetivos, los riesgos, las limitaciones, la criticidad, la testabilidad y la disponibilidad de recursos. A medida que la planificación del proyecto y de las pruebas van avanzando, habrá más información disponible y el plan podrá incluir más detalles.

La planificación de pruebas es una actividad continua que se lleva a cabo en todos los procesos de ciclo de vida y actividades. El feedback de las actividades de pruebas sirve para reconocer los riesgos cambiantes con vistas a ajustar la planificación.

5.2.2 Actividades de planificación de pruebas (K3)

Las actividades de planificación para un sistema completo o para parte de un sistema pueden incluir:

- Determinar el alcance y los riesgos e identificar los objetivos de las pruebas.
- Definir el enfoque global de las pruebas, incluida la definición de los niveles de pruebas y los criterios de entrada y salida.
- Integrar y coordinar las actividades de pruebas en las actividades de ciclo de vida del software (adquisición, suministro, desarrollo, operación y mantenimiento).
- Adoptar decisiones sobre qué probar, qué cargos llevarán a cabo las actividades de pruebas, cómo deberían realizarse las actividades de pruebas y cómo se evaluarán los resultados de las pruebas.
- Programar las actividades de análisis y diseño de las pruebas.
- Programar la implementación, ejecución y evaluación de las pruebas.
- Asignar recursos para las distintas actividades definidas.
- Definir la cantidad, el nivel de detalle, la estructura y las plantillas para la documentación de las pruebas.

- Seleccionar métricas para el seguimiento y el control de la preparación y ejecución de las pruebas, la resolución de defectos y los temas de riesgos.
- Establecer el nivel de detalle de los procedimientos de prueba para facilitar información suficiente para dar soporte a la preparación y ejecución reproducible de pruebas.

5.2.3 Criterios de entrada (K2)

Los criterios de entrada establecen cuándo iniciar las pruebas, por ejemplo al inicio de un nivel de prueba o cuando una serie de pruebas esté lista para su ejecución.

En general, los criterios de entrada pueden cubrir lo siguiente:

- Disponibilidad y disposición del entorno de pruebas.
- Disposición de las herramientas de prueba en el entorno de pruebas.
- Disponibilidad de código testeable.
- Disponibilidad de datos de prueba.

5.2.4 Criterios de salida (K2)

Los criterios de salida establecen cuándo detener las pruebas, como por ejemplo al final de un nivel de prueba o cuando una serie de pruebas haya logrado un objetivo específico.

En general, los criterios de salida pueden cubrir lo siguiente:

- Medidas de exhaustividad, tales como la cobertura de código, la funcionalidad o el riesgo.
- Estimaciones de densidad de defectos o medidas de fiabilidad.
- Coste
- Riesgos residuales, tales como defectos no corregidos o ausencia de Cobertura de pruebas en determinadas áreas.
- Calendarios, como los basados en el plazo de comercialización.

5.2.5 Estimación de pruebas (K2)

Dos enfoques distintos para estimar el esfuerzo de pruebas son:

- El enfoque basado en métricas: estimar el esfuerzo de pruebas en base a métricas de proyectos anteriores o similares o en base a valores típicos.
- El enfoque basado en expertos: estimar las tareas en base a las estimaciones hechas por el propietario de las tareas o por los expertos.

Una vez estimado el esfuerzo de prueba, pueden identificarse los recursos y elaborarse un calendario.

El esfuerzo de prueba puede depender de una serie de factores, entre los que se incluyen:

- Las características del producto: la calidad de la especificación y demás información utilizada para modelos de prueba (es decir, la base de prueba), el tamaño del producto, la complejidad del dominio del problema, los requisitos de fiabilidad y seguridad y los

requisitos de documentación.

- Las características del proceso de desarrollo: la estabilidad de la organización, las herramientas utilizadas, el proceso de pruebas, las habilidades de las personas implicadas y la presión temporal.
- El resultado de las pruebas: el número de defectos y la cantidad de cambios necesarios.

5.2.6 Estrategia de pruebas, enfoque de pruebas (K2)

El enfoque de pruebas es la aplicación de la estrategia de pruebas para un proyecto específico. El enfoque de pruebas se define y redefine en los planes y diseños de prueba. Generalmente, incluye las decisiones tomadas en base al objetivo y a la evaluación de los riesgos del proyecto (de prueba). Constituye el punto de inicio para planificar el proceso de pruebas, para seleccionar las técnicas de diseño de pruebas y los tipos de pruebas a aplicar, y para definir los criterios de entrada y salida.

El enfoque seleccionado dependerá del contexto y puede tener en cuenta los riesgos, los peligros y la seguridad, los recursos disponibles y habilidades, la tecnología, la naturaleza del sistema (por ejemplo, personalizado vs COTS), los objetivos de prueba y la normativa.

Entre los enfoques típicos se incluyen:

- Enfoques analíticos, como las pruebas basadas en riesgos en las que las pruebas se concentran en las áreas de mayor riesgo.
- Enfoques basados en modelos, como las pruebas estocásticas que utilizan información estadística sobre frecuencias de fallos (tales como los modelos de fiabilidad de crecimiento) o uso (tales como perfiles operativos).
- Enfoques metódicos, como los basados en fallos (incluyendo la predicción de errores y los ataques de faltas) los basados en la experiencia, los basados en listas de comprobación y los basados en características de calidad.
- Los enfoques de proceso – o enfoques ajustados a las normas, como los especificados en las normas específicas del sector o en las distintas metodologías ágiles.
- Enfoques dinámicos y heurísticos, como las pruebas exploratorias en las que las pruebas son más reactivas a eventos en lugar de estar planificadas con antelación, y en las que las tareas de ejecución y evaluación son simultáneas.
- Enfoques consultivos, como aquellos en los que la Cobertura de pruebas se establece principalmente en base a las recomendaciones y orientaciones de los expertos de tecnología y/o negocio más allá del equipo de pruebas.
- Enfoques anti-regresión, como los que incluyen la reutilización del material de pruebas existente, la automatización extensiva de pruebas de regresión funcionales y juegos de prueba estándar.

Pueden combinarse varios enfoques y adoptar, por ejemplo, un enfoque dinámico basado en el riesgo.

5.3 Seguimiento y control del progreso de las pruebas (K2)	*20 minutos*

Términos

Densidad de defectos, frecuencia de fallos, seguimiento de pruebas, informe resumen de pruebas

5.3.1 Seguimiento del progreso de las pruebas (K1)

El objetivo del seguimiento de las pruebas es facilitar feedback y visibilidad sobre las actividades de pruebas. La información a controlar puede recopilarse de forma manual o automática y puede utilizarse para medir los criterios de salida, tales como la cobertura. Las métricas también pueden utilizarse para evaluar el progreso contra el programa y el presupuesto previstos. Entre las métricas de prueba más comunes se incluyen:

- Porcentaje de trabajo realizado durante la preparación del caso de prueba (o porcentaje de casos de prueba planificados preparados).
- Porcentaje de trabajo realizado durante la preparación del entorno de pruebas.
- Ejecución del caso de prueba (por ejemplo, número de casos de prueba ejecutados/no ejecutados, y casos de prueba superados/fallidos).
- Información sobre defectos (por ejemplo, densidad de los defectos, defectos detectados y corregidos, frecuencia de fallos y resultados de la repetición de pruebas).
- Cobertura de pruebas de los requisitos, riesgos o código.
- Confianza subjetiva de los probadores en el producto.
- Fechas de hitos de las pruebas.
- Coste de las pruebas, incluyendo el coste comparado con el beneficio de detectar el siguiente defecto o de ejecutar la siguiente prueba.

5.3.2 Informes de pruebas (K2)

Los informes de pruebas tienen por objeto resumir información sobre el entorno de pruebas, incluyendo:

- Qué ha pasado durante un período de prueba, como fechas en las que se cumplieron los criterios de salida.
- Análisis de información y métricas para respaldar las recomendaciones y decisiones sobre acciones futuras, tales como una evaluación de los defectos restantes, el beneficio económico de las pruebas continuadas, los riesgos pendientes y el nivel de confianza en el software probado.

El diseño de un informe resumen de pruebas se aborda en la “Norma para la documentación

de prueba de software" (IEEE Std 829-1998).

Las métricas deben recopilarse durante y al final de cada nivel de prueba con vistas a evaluar:

- La idoneidad de los objetivos de la prueba para dicho nivel de prueba.
- La idoneidad de los enfoques de prueba adoptados.
- La efectividad de las pruebas por lo que respecta a los objetivos.

5.3.3 Control de pruebas (K2)

El control de pruebas describe cualquier acción orientativa o correctiva adoptada como resultado de la información y las métricas recopiladas e incluidas en un informe. Las acciones pueden referirse a cualquier actividad de prueba y pueden afectar a cualquier actividad o tarea del ciclo de vida del software.

Algunos ejemplos de acciones de control de pruebas incluyen:

- Tomar decisiones en base a información obtenida del seguimiento de las pruebas.
- Restablecer la prioridad de las pruebas si sucede un riesgo identificado (por ejemplo, demora en la entrega del software).
- Cambiar el calendario de pruebas por motivos de disponibilidad o no disponibilidad de un entorno de pruebas.
- Establecer un criterio de entrada que requiera la repetición de las pruebas de las correcciones (pruebas de confirmación) por parte de un desarrollador antes de aceptarlas en una construcción

5.4 Gestión de la configuración (K2)	*10 minutos*

Términos

Gestión de la configuración, control de versión

Antecedentes

El objetivo de la gestión de la configuración es establecer y mantener la integridad de los productos (componentes, datos y documentación) del software o sistema a través del ciclo de vida del proyecto o del producto.

En el caso de las pruebas, la gestión de la configuración puede implicar garantizar lo siguiente:

- Todos los elementos de soporte de prueba han sido identificados, se ha llevado a cabo un control de versión, se han localizado los cambios, tanto entre sí como en relación con los elementos de desarrollo (objetos de prueba) de manera que puede mantenerse la trazabilidad a lo largo de todo el proceso de pruebas.
- Todos los documentos y elementos de software identificados están referenciados de manera clara en la documentación de prueba

Para el probador, la gestión de la configuración le ayuda a identificar (y reproducir) de manera unívoca el elemento probado, los documentos de prueba, las pruebas y los arneses de pruebas.

Durante la planificación de las pruebas, deben seleccionarse, documentarse e implementarse los procedimientos de gestión de la configuración y las (herramientas) de infraestructura.

5.5 Riesgos y pruebas (K2)	*30 minutos*

Términos

Riesgo de producto, riesgo de proyecto, pruebas basadas en riesgos

Antecedentes

El riesgo puede definirse como la oportunidad de un evento, peligro, amenaza o situación que sucede y tiene como resultados consecuencias no deseadas o un problema potencial. El nivel de riesgo vendrá determinado por la probabilidad de que ocurra un evento adverso y su impacto (el daño resultante de dicho evento).

5.5.1 Riesgos de proyecto (K2)

Los riesgos de proyecto son los riesgos relativos a la capacidad del proyecto de lograr sus objetivos, tales como:

- Factores de organización:
  - Aptitudes, formación y falta de personal
  - Aspectos de personal
  - Aspectos políticos, tales como:
    - Problemas con la forma en la que los probadores comunican sus necesidades y los resultados de las pruebas.
    - Incapacidad del equipo de hacer un seguimiento de la información encontrada durante las pruebas y revisiones (por ejemplo, no se mejora el desarrollo y las prácticas de pruebas).
  - Actitud indebida o falsas expectativas ante las pruebas (por ejemplo, no valorar el valor de detectar defectos durante las pruebas).
- Aspectos técnicos:
  - Problemas para definir los requisitos adecuados.
  - La medida en que no pueden cumplirse los requisitos dadas las limitaciones existentes.
  - El entorno de pruebas no está listo a tiempo.
  - Demora en la conversión de datos, planificación y desarrollo de migración y conversión de datos de prueba/herramientas de migración.
  - Baja calidad del diseño, código, datos de configuración, datos de prueba y pruebas.
- Aspectos de proveedores:

- Fallos de terceros
- Aspectos contractuales

A la hora de analizar, gestionar y mitigar estos riesgos, el jefe de pruebas debe seguir principios de gestión de proyectos bien establecidos. La "Norma de Documentación de prueba de Software" (Norma IEEE 829-1998) esboza los planes de pruebas y exige que se indiquen los riesgos y contingencias.

5.5.2 Riesgos de producto (K2)

Las posibles áreas de fallo (eventos futuros adversos o peligros) en el software o sistema se conocen como riesgos de producto, ya que suponen un riesgo para la calidad del producto. Entre dichos riesgos se encuentran:

- El software entregado es proclive a los fallos
- La posibilidad de que el software/hardware pueda dañar a un individuo o a una empresa.
- Malas características del software (por ejemplo, funcionalidad, fiabilidad, usabilidad y rendimiento).
- Mala integridad y calidad de los datos (por ejemplo, problemas de migración de datos, problemas de conversión de datos, problemas de transporte de datos, violación de estándares de datos).
- El software no realiza las funciones previstas.

Los riesgos sirven para decidir dónde empezar las pruebas y dónde probar más; las pruebas sirven para reducir el riesgo de que suceda un efecto adverso, o para reducir el impacto del mismo.

Los riesgos de producto constituyen un tipo especial de riesgo para el éxito de un proyecto. El hecho de realizar pruebas a modo de actividad de control del riesgo proporciona feedback sobre el riesgo residual midiendo la efectividad de la eliminación de los defectos críticos y los planes de contingencia.

El enfoque basado en el riesgo en las pruebas ofrece oportunidades proactivas de reducir los niveles de riesgo de producto, empezando en las etapas iniciales de un proyecto. Este enfoque implica la identificación de riesgos de producto y su uso en la planificación de las pruebas y especificación de control, preparación y ejecución de las pruebas. En un enfoque basado en el riesgo, los riesgos identificados pueden utilizarse para:

- Establecer las técnicas de pruebas a emplear.
- Establecer el alcance de las pruebas a ejecutar.
- Priorizar las pruebas en un intento por identificar los defectos críticos lo antes posible.
- Establecer si podría utilizarse alguna actividad no de prueba para reducir el riesgo (por ejemplo, impartir formación a diseñadores sin experiencia).

Las pruebas basadas en riesgos se basan en el conocimiento colectivo y en la comprensión

de las partes interesadas del proyecto para establecer los riesgos y los niveles de pruebas necesarios para abordar dichos riesgos.

Con vistas a garantizar que la posibilidad de fallo de un producto es mínima, las actividades de gestión del riesgo establecen un enfoque disciplinando para:

- Evaluar (y re-evaluar de manera regular) qué puede fallar (riesgos).
- Establecer qué riesgos son importante tratar.
- Implementar acciones para abordar dichos riesgos.

Además, las pruebas pueden contribuir a identificar nuevos riesgos, pueden ayudar a establecer qué riesgos deben reducirse y pueden reducir la incertidumbre sobre los riesgos.

5.6 Gestión de incidencias (K3)	*40 minutos*

Términos

Registro de incidencias, gestión de incidencias, informe de incidencias

Antecedentes

Dado que uno de los objetivos de las pruebas es la detección de defectos, las discrepancias entre los resultados reales y los resultados esperados deben registrarse como incidencias. Todas las incidencias deben ser investigadas, ya que algunas pueden constituir defectos. A continuación, se definen las acciones adecuadas para eliminar las incidencias y los defectos. Las incidencias y los defectos se rastrean desde su identificación y clasificación hasta la corrección y confirmación de su solución. Con vistas a gestionar todas las incidencias hasta su compleción, la organización, debe establecer un proceso de gestión de incidencias y normas para su clasificación.

Las incidencias pueden surgir durante el desarrollo, la revisión, las pruebas o el uso de un producto de software. Asimismo, pueden surgir por problemas en el código o en el sistema de funcionamiento, o en cualquier tipo de documentación que incluya requisitos, documentos de desarrollo, documentos de prueba e información de usuario, como guías de ayuda o instalación.

Los informes de incidencias tienen los siguientes objetivos:

- Facilitar feedback sobre el problema a los desarrolladores y demás partes implicadas permitiendo su identificación, aislamiento y corrección, según proceda.
- Proporcionar a los líderes de prueba un medio de seguimiento de la calidad del sistema probado y del progreso de las pruebas.
- Aportar ideas para la mejora del proceso de pruebas.

El informe de incidencias puede incluir los siguientes detalles:

- Fecha de expedición, organización emisora y autor.
- Resultados esperados y resultados reales.
- Identificación del elemento de prueba (elemento de configuración) y entorno.
- Proceso del ciclo de vida del software o del sistema en el que se ha observado la incidencia.
- Descripción de la incidencia para permitir su reproducción y resolución, incluyendo registros, volcados de bases de datos o pantallazos.
- Alcance o grado de impacto en los intereses de las partes interesadas.

- Gravedad del impacto en el sistema.
- Urgencia/prioridad de su corrección.
- Estado de la incidencia (por ejemplo, abierta, diferida, duplicada, esperando corrección, corregida esperando la repetición de las pruebas, cerrada).
- Conclusiones, recomendaciones y autorizaciones.
- Aspectos globales, como otras áreas que pueden verse afectadas por un cambio derivado de la incidencia.
- Historial del cambio, como la secuencia de acciones adoptadas por los miembros del equipo de proyecto por lo que respecta a la incidencia para aislarla, repararla y confirmarla como corregida.
- Referencias, incluyendo la identidad de la especificación de caso de prueba que puso de manifiesto el problema.

La estructura de los informes de incidencias se aborda en la "Norma para la documentación de prueba de software" (IEEE Std 829-1998).

Referencias

5.1.1 Black, 2001, Hetzel, 1988

5.1.2 Black, 2001, Hetzel, 1988

5.2.5 Black, 2001, Craig, 2002, IEEE Std 829-1998, Kaner 2002

5.3.3 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE Std 829-1998

5.4 Craig, 2002

5.5.2 Black, 2001, IEEE Std 829-1998

5.6 Black, 2001, IEEE Std 829-1998

6. Herramientas de soporte de pruebas (K2)	80 minutos

Objetivos de aprendizaje de Herramientas de soporte de pruebas

Los objetivos identifican lo que será usted capaz de hacer tras la compleción del módulo.

6.1 Tipos de herramientas de pruebas (K2)

LO-6.1.1 Clasificar distintos tipos de herramientas de pruebas en función de su objetivo y de las actividades del proceso de pruebas fundamental y del ciclo de vida del software (K2).

LO-6.1.3 Explicar el término herramienta de pruebas y el objetivo de las herramientas de soporte de pruebas (K2).

6.2 Uso efectivo de las herramientas: Posibles ventajas y riesgos (K2)

LO-6.2.1 Resumir las posibles ventajas y riesgos de la automatización de las pruebas y de las herramientas de soporte de pruebas (K2).

LO-6.2.2 Recordar consideraciones particulares sobre las herramientas de ejecución de pruebas, los análisis estáticos y las herramientas de gestión de pruebas (K1).

6.3 Introducción de una herramienta en una organización (K1)

LO-6.3.1 Establecer los principios básicos de la introducción de una herramienta en una organización (K1).

LO-6.3.2 Establecer los objetivos de una prueba de concepto para la evaluación de herramientas y una fase piloto para la implementación de herramientas.

LO-6.3.3 Reconocer qué factores, además de adquirir la herramienta, son necesarios para obtener una buena herramienta de soporte (K1).

6.1 Tipos de herramientas de pruebas (K2)	*45 minutos*

Términos

Herramienta de gestión de la configuración, herramienta de cobertura, herramienta de depuración, herramienta de análisis dinámico, herramienta de gestión de incidencias, herramienta de pruebas de carga, herramienta de modelado, herramienta de monitorización, herramienta de pruebas de rendimiento, efecto sonda, herramienta de gestión de requisitos, herramienta de revisión, herramienta de seguridad, herramienta de análisis estático, herramienta de pruebas de estrés, comparador de pruebas, herramienta de

preparación de datos de prueba, herramienta de diseño de pruebas, arnés de prueba, herramienta de ejecución de pruebas, herramienta de gestión de pruebas, herramienta de marco de trabajo de pruebas unitarias

6.1.1 Significado y objetivo de las herramientas de soporte de pruebas (K2)

Las herramientas de pruebas pueden utilizarse en una o más actividades de soporte de pruebas. Entre las que se encuentran:

1. Las herramientas que se utilizan directamente en las pruebas, como las herramientas de ejecución de pruebas, las herramientas de generación de datos de prueba y las herramientas de comparación de resultados.
2. Las herramientas que ayudan a gestionar el proceso de pruebas, como las que sirven para gestionar pruebas, resultados de pruebas, datos, requisitos, incidencias, defectos, etc., y para elaborar informes y monitorizar la ejecución de pruebas.
3. Las herramientas que se utilizan en la fase de reconocimiento, o en otras palabras: exploración (por ejemplo, herramientas que monitorizan la actividad de archivos de una aplicación).
4. Cualquier herramienta que contribuye al proceso de pruebas (en este sentido, una hoja de datos también se considera una herramienta de pruebas).

Las herramientas de soporte de pruebas pueden tener uno o más de los siguientes objetivos, en función del contexto:

- Mejorar la eficiencia de las tareas de pruebas automatizando tareas repetitivas o dando soporte a las actividades de pruebas manuales, como la planificación, el diseño, la elaboración de informes y la monitorización de pruebas
- Automatizar aquellas actividades que requieren muchos recursos si se hacen de forma manual (como por ejemplo, las pruebas estáticas).
- Automatizar aquellas actividades que no pueden ejecutarse de forma manual (como por ejemplo, pruebas de rendimiento a gran escala de aplicaciones cliente-servidor).
- Aumentar la fiabilidad de las pruebas (por ejemplo, automatizando las comparaciones de grandes ficheros de datos y simulando comportamientos).

El término “marcos de trabajo de pruebas” se utiliza a menudo en el sector y puede tener, como mínimo, cualquiera de los tres siguientes significados:

- Librerías de pruebas reutilizables y ampliables que pueden utilizarse para crear herramientas de pruebas (también conocido como arnés de pruebas).
- Un tipo de diseño de automatización de pruebas (por ejemplo, guiadas por datos o guiadas por palabras clave).
- Proceso general de ejecución de las pruebas.

A efectos de este, el término “marcos de trabajo de pruebas” se utiliza en sus dos primeros significados, según lo descrito en la Sección 6.1.6.

6.1.2 Clasificación de herramientas de prueba (K2)

Hay varias herramientas que dan soporte a distintos aspectos de las pruebas. Las herramientas pueden clasificarse en base a distintos criterios, tales como el objetivo, comercial / libre / fuente abierta / "shareware", tecnología utilizada, etc. En este programa de estudio, las herramientas se clasifican en función de las actividades de pruebas a las que dan soporte.

Algunas herramientas dan soporte a una actividad de manera clara, mientras que otras pueden dar soporte a más de una actividad, pero se clasifican dentro de la actividad a la que están más estrechamente vinculadas. Las herramientas procedentes de un único proveedor, especialmente aquellas que han sido diseñadas para funcionar juntas, pueden incluirse en un solo paquete.

Algunos tipos de herramientas de prueba pueden ser intrusivos, es decir, pueden afectar al resultado real de la prueba. Así por ejemplo, los tiempos reales pueden diferir debido a las instrucciones adicionales que la herramienta ejecuta, o incluso puede obtenerse una medida distinta de cobertura de código. La consecuencia del uso de herramientas intrusivas se denomina efecto sonda.

Algunas herramientas ofrecen un soporte más adecuado para los desarrolladores (como por ejemplo, las herramientas que se utilizan durante las pruebas de componente y de integración de componentes). Estas herramientas aparecen marcadas con el símbolo “(D)” en la lista a continuación.

6.1.3 Herramientas de soporte para la gestión de pruebas (K1)

Las herramientas de gestión son aplicables a todas las actividades de pruebas durante todo el ciclo de vida del software.

Herramientas de gestión de pruebas

Estas herramientas ofrecen interfaces para ejecutar pruebas, localizar defectos y gestionar requisitos, además de dar soporte al análisis cuantitativo y a la elaboración de informes sobre los objetos de prueba. Asimismo ayudan a localizar los objetos de prueba conforme a las especificaciones de requisitos y pueden tener una capacidad de control de versión independiente o una interfaz conforme a una externa.

Herramientas de gestión de requisitos

Estas herramientas almacenan sentencias de requisitos, almacenan los atributos de los requisitos (incluida la prioridad), proporcionan identificadores únicos y facilitan la localización de los requisitos para pruebas individuales. Asimismo, estas herramientas pueden ayudar a identificar la ausencia o la inconsistencia de requisitos.

Herramientas de gestión de incidencias (Herramientas de seguimiento de defectos)

Estas herramientas almacenan y gestionan informes de incidencias, es decir defectos, fallos,

cambio de peticiones o problemas y anomalías percibidas, y ayudan a gestionar el ciclo de vida de las incidencias, y alternativamente contribuyen al análisis estático.

Herramientas de gestión de la configuración

A pesar de no constituir estrictamente herramientas de prueba, estas herramientas son necesarias para el almacenamiento y la gestión de versiones de productos de soporte de software y software asociado, especialmente a la hora de configurar más de un entorno de hardware/software en términos de versiones del sistema operativo, compiladores, navegadores, etc.

6.1.4 Herramientas de soporte para las pruebas estáticas (K1)

Las herramientas de prueba estática proporcionan una manera efectiva de localizar más defectos en una etapa más temprana del proceso de desarrollo.

Herramientas de revisión

Estas herramientas son útiles en los procesos de revisión, listas de comprobación y directrices de revisión, y se utilizan para almacenar y comunicar comentarios de revisión, informes sobre defectos y esfuerzos. Asimismo, pueden servir de ayuda para las revisiones en línea de equipos grandes o geográficamente dispersos.

Herramientas de análisis estático (D)

Estas herramientas ayudan a los desarrolladores y probadores a localizar defectos antes de realizar pruebas dinámicas proporcionándoles soporte para aplicar las normas de codificación (incluyendo la codificación segura), el análisis de las estructuras y las dependencias. Asimismo, pueden ser útiles para planificar o analizar los riesgos facilitando métricas para el código (por ejemplo, complejidad).

Herramientas de modelado (D)

Estas herramientas se utilizan para validar modelos de software (tales como, modelo de datos físicos (PDM) de una base de datos relacional) enumerando inconsistencias y localizando defectos. Estas herramientas a menudo sirven para generar algunos casos de prueba basados en un modelo.

6.1.5 Herramientas de soporte para la especificación de prueba (K1)

Herramientas de diseño de pruebas

Estas herramientas sirven para generar entradas de prueba o pruebas ejecutables y/o oráculos de prueba a partir de requisitos, interfaces gráficas de usuario, modelos de diseño (estado, dato u objeto) o códigos.

Herramientas de preparación de datos de prueba

Las herramientas de preparación de datos de prueba manejan bases de datos, archivos o transmisiones de datos para configurar los datos de prueba que se utilizan durante la ejecución de pruebas con vistas a garantizar la seguridad a través del anonimato de los datos.

6.1.6 Herramientas de soporte para la ejecución y el registro de pruebas (K1)

Herramientas de ejecución de pruebas

Estas herramientas permiten ejecutar pruebas de manera automática, o semiautomática, utilizando entradas almacenadas y resultados esperados, a través del uso de un lenguaje de creación de scripts y generalmente ofrecen un registro de prueba para cada ejecución de una prueba. Asimismo, pueden servir para registrar pruebas. Normalmente soportan lenguaje de creación de scripts o una configuración basada en GUI para la parametrización de datos y demás personalización durante las pruebas.

Arnés de pruebas/Herramientas de marco de trabajo de pruebas unitarias (D)

El arnés de pruebas unitarias o el marco de trabajo facilitan el proceso de pruebas en componentes o partes de un sistema simulando el entorno en el que se ejecutará dicho objeto de prueba, a través de la creación de objetos de imitación como "stubs" o controladores.

Comparadores de pruebas

Los comparadores de pruebas establecen las diferencias entre archivos, bases de datos o resultados de pruebas. Las herramientas de ejecución de pruebas normalmente incluyen comparadores dinámicos, pero la comparación posterior a la ejecución debe hacerse con una herramienta de comparación aparte. Los comparadores de pruebas pueden utilizar oráculos de pruebas, especialmente si están automatizados.

Herramientas de medición de la cobertura (D)

Estas herramientas, a través de medios intrusivos o no intrusivos, miden el porcentaje de tipos específicos de estructuras de código que ha sido practicado (por ejemplo, sentencias, ramas o decisiones y llamadas a módulos o funciones) por parte de una serie de probadores.

Herramientas de pruebas de seguridad

Estas herramientas sirven para evaluar las características de seguridad del software, entre las que se incluyen evaluar la capacidad del software para proteger la confidencialidad, integridad, autenticación, autorización, disponibilidad y no repudiación de los datos. Las herramientas de seguridad se centran principalmente en una tecnología, una plataforma y un alcance específicos.

6.1.7 Herramientas de soporte para el rendimiento y la monitorización (K1)

Herramientas de análisis dinámico (D)

Las herramientas de análisis dinámico localizan defectos que pasan a ser evidentes sólo una vez que el software está ejecutándose, tales como las dependencias de tiempo o las pérdidas de memoria. Normalmente se utilizan en las pruebas de componente e integración de componentes y en las pruebas de "middleware".

Herramientas de pruebas de rendimiento/pruebas de carga/pruebas de estrés

Las herramientas de pruebas de rendimiento monitorizan y reportan sobre el comportamiento de un sistema en una serie de condiciones de uso simuladas en términos de usuarios simultáneos, modelo de lanzamiento, frecuencia y porcentaje asociado de transacciones. La simulación de carga se logra creando usuarios virtuales que llevan a cabo una serie seleccionada de transacciones, esparcidos en distintas máquinas de prueba conocidos comúnmente como generadores de carga.

Herramientas de monitorización

Las herramientas de monitorización analizan, comprueban y reportan continuamente el uso de recursos específicos del sistema y lanzan avisos sobre posibles problemas de servicio.

6.1.8 Herramientas de soporte para necesidades de pruebas específicas (K1)

Evaluación de la calidad de los datos

Los datos son la pieza clave de muchos proyectos, como los proyectos de conversión/migración, y las aplicaciones, tales como los data "warehouses" y sus atributos, pueden variar en cuanto a criticidad y volumen. En estos contextos, las herramientas deben emplearse para evaluar la calidad con vistas a revisar y comprobar las reglas de conversión y migración de datos para garantizar que los datos procesados son correctos, completos y se ajustan a una norma predefinida específica del contexto.

También hay otras herramientas de prueba disponibles para probar la usabilidad.

6.2 Uso efectivo de las herramientas: Ventajas potenciales y riesgos (K2)	*20 minutos*

Términos

Pruebas guiadas por datos, pruebas guiadas por palabra clave, lenguaje de creación de scripts

6.2.1 Ventajas potenciales y riesgos de las herramientas de soporte de pruebas (para todas las herramientas) (K2)

El mero hecho de comprar o arrendar una herramienta no nos garantiza el éxito con dicha herramienta. Cada tipo de herramienta puede exigir un esfuerzo adicional para lograr ventajas reales y duraderas. El uso de herramientas plantea posibles ventajas y oportunidades, pero también hay riesgos.

Entre las posibles ventajas de utilizar herramientas se encuentran las siguientes:

- Reducción del trabajo repetitivo (por ejemplo, la ejecución de pruebas de regresión, la reintroducción de los mismos datos de prueba y la comprobación contra estándares de codificación).
- Mayor consistencia y respetabilidad (por ejemplo, las pruebas ejecutadas por una herramienta en el mismo orden y con la misma frecuencia, y pruebas derivadas de los requisitos).
- Evaluación de los objetivos (por ejemplo, medidas estáticas, cobertura).
- Facilidad de acceso a la información sobre las pruebas (por ejemplo, estadísticas y gráficos sobre el avance de las pruebas, la frecuencia de incidencias y el rendimiento).

Entre los riesgos de utilizar herramientas se encuentran:

- Expectativas poco realistas de la herramienta (incluyendo funcionalidad y facilidad de uso).
- Subestimación de la cantidad de tiempo, coste y esfuerzo necesario para la introducción inicial de una herramienta (incluyendo formación y experiencia externa).
- Subestimación del tiempo y el esfuerzo necesarios para conseguir ventajas significativas y constantes de la herramienta (incluyendo la necesidad de cambios en el proceso de pruebas y la mejora continua de la forma en la que se utiliza la herramienta).
- Subestimación del esfuerzo necesario para mantener los activos de prueba generados por la herramienta.
- Exceso de confianza en la herramienta (sustitución del diseño de pruebas o uso de pruebas automatizadas cuando sería mejor llevar a cabo pruebas manuales).

- Desprecio del control de versión de los activos de prueba en la herramienta.
- Desprecio de problemas de relaciones e interoperabilidad entre herramientas críticas, tales como las herramientas de gestión de requisitos, herramientas de control de versiones, herramientas de gestión de incidencias, herramientas de seguimiento de defectos y herramientas procedentes de varios fabricantes.
- Riesgo de que el fabricante de la herramienta cierre, retire la herramienta o venda la herramienta a otro proveedor.
- Mala respuesta del fabricante para soporte, actualizaciones y corrección de defectos.
- Riesgo de suspensión de proyectos de código abierto/sin herramientas.
- Imprevistos, tales como la incapacidad de soportar una nueva plataforma.

6.2.2 Consideraciones especiales para algunos tipos de herramientas (K1)

Herramientas de ejecución de pruebas

Las herramientas de ejecución de pruebas ejecutan objetos de prueba sirviéndose de guiones de prueba automatizados. Este tipo de herramienta a menudo requiere un esfuerzo importante para lograr ventajas significativas.

El hecho de capturar las pruebas registrando las acciones de un probador manual puede parecer atractivo, pero este enfoque no es escalable a grandes cantidades de guiones de prueba automatizados. Un guión capturado es una representación lineal que incluye datos y acciones específicos como parte de cada guión. Este tipo de guión puede resultar inestable si se producen eventos imprevistos.

Un enfoque de pruebas guiadas por datos separa las entradas de las pruebas (los datos), normalmente en una hoja de datos, y se sirve de un script de prueba más genérico que pueda leer los datos de entrada y ejecutar el mismo script de prueba con distintos datos. Los probadores que no están familiarizados con el lenguaje de creación de scripts de este modo pueden crear los datos de prueba para estos scripts predefinidos.

Asimismo, en las técnicas guiadas por datos se utilizan otras técnicas en las que, en lugar de colocar en una hoja de datos combinaciones de datos codificadas, los datos se general utilizando algoritmos en base a parámetros configurables en el momento de la ejecución y facilitados a la aplicación. Así por ejemplo, una herramienta puede utilizar un algoritmo que genere una identificación de usuario aleatoria, y a efectos de repetibilidad en modelo, se utiliza una siembra para controlar la aleatoriedad.

En un enfoque de pruebas guiado por palabras clave, la hoja de datos contiene las palabras clave que describen las acciones a adoptar (también conocidas como palabras de acción) y datos de prueba. Los probadores (a pesar de no estar familiarizados con el lenguaje de creación de scripts) pueden entonces definir pruebas utilizando las palabras clave, las cuales pueden ajustarse a la aplicación en pruebas.

Para todos los enfoques es necesario contar con experiencia técnica en el ámbito del

lenguaje de creación de scripts (tanto por parte de los probadores como por parte de los especialistas en la automatización de pruebas).

Independientemente de la técnica de "scripting" utilizada, los resultados esperados para cada prueba deben almacenarse para su posterior comparación.

Herramientas de análisis estático

Las herramientas de análisis estático aplicadas al código fuente pueden aplicar estándares de codificación, pero si se aplican a un código existente, pueden generar una gran cantidad de mensajes. Los mensajes de advertencia no impiden que el código se traduzca en un programa ejecutable, pero idealmente deberían tratarse de manera que faciliten el mantenimiento del código en el futuro. La implementación gradual de la herramienta de análisis con filtros iniciales para excluir ciertos mensajes constituye un enfoque efectivo.

Herramientas de gestión de pruebas

Las herramientas de gestión de pruebas tienen que interactuar con otras herramientas u hojas de datos para producir información útil en un formato que se ajuste a las necesidades de la organización.

6.3 Introducción de una herramienta en una organización (K1)	*15 minutos*

Términos

Ningún término específico.

Antecedentes

Entre las principales consideraciones a tener en cuenta a la hora de seleccionar una herramienta para una organización se encuentran:

- Análisis de la madurez organizativa, fortalezas y debilidades e identificación de las oportunidades para un proceso de pruebas mejorado soportado por herramientas.
- Evaluación frente a requisitos claros y criterios objetivos.
- Una prueba de concepto, utilizando una herramienta de prueba durante la fase de evaluación para establecer si rinde de manera eficiente con el software objeto de la prueba y dentro de la actual infraestructura o para identificar los cambios necesarios en dicha infraestructura para utilizar la herramienta de manera efectiva.
- Evaluación del fabricante (incluyendo formación, soporte y aspectos comerciales) o soporte de servicio a proveedores en caso de herramientas no comerciales.
- Identificación de requisitos internos para impartir "coaching" y formación sobre el uso de la herramienta.
- Evaluación de las necesidades de formación habida cuenta de las habilidades de automatización de pruebas del equipo de pruebas.
- Cálculo de un ratio coste-beneficio en base a un caso de negocio concreto.

Para introducir la herramienta seleccionada en una organización debe lanzarse un proyecto piloto, cuyos objetivos son los siguientes:

- Aprender más detalles sobre la herramienta.
- Evaluar la forma en la que la herramienta se ajusta a los procesos y prácticas existentes, y determinar qué debe cambiar.
- Decidir las formas estándar en las que se debe utilizar, gestionar, almacenar y mantener la herramienta y los activos de prueba (por ejemplo, decidir sobre la designación de archivos y pruebas, la creación de bibliotecas y definir la modularidad de los juegos de pruebas).
- Valorar si se lograrán los beneficios a un coste razonable.

Los factores de éxito para el despliegue de la herramienta dentro de una organización incluyen:

- Hacer extensiva la herramienta al resto de la organización de una manera incremental.
- Adaptar y mejorar los procesos para adaptarlos al uso de la herramienta.
- Impartir formación y "coaching"/tutorías a los nuevos usuarios.
- Definir las directrices de uso.
- Aplicar una forma de recopilar información sobre el uso de la herramienta a partir de su uso real.
- Hacer un seguimiento del uso y de los beneficios de la herramienta.
- Proporcionar soporte al equipo de pruebas para una herramienta dada.
- Recopilar las lecciones aprendidas de todos los equipos.

Referencias

6.2.2 Buwalda, 2001, Fewster, 1999

6.3 Fewster, 1999

7. Referencias

Normas

Glosario de términos utilizados en pruebas de software del ISTQB, Versión 2.1

[CMMI] Chrissis, M.B., Konrad, M. and Shrum, S. (2004) CMMI, Guidelines for Process Integration *and Product Improvement*, Addison Wesley: Reading, MA
Véase la sección 2.1
[IEEE Std 829-1998] Norma IEEE 829TM (1998) Norma IEEE para documentación de software.
Véase las secciones 2.3, 2.4, 4.1, 5.2, 5.3, 5.5, 5.6

[IEEE 1028] Norma IEEE 1028TM (2008) Norma IEEE para revisiones y auditorías de software.
Véase la sección 3.2

[IEEE 12207] IEEE 12207/ISO/IEC 12207-2008, Procesos del ciclo de vida del software. Véase la sección 2.1

[ISO 9126] ISO/IEC 9126-1:2001, Ingeniería de software - Calidad del producto de Software.
Véase la sección 2.3

Libros

[Beizer, 1990] Beizer, B. (1990) Software Testing Techniques (2nd edition), Van Nostrand Reinhold: Boston
Véanse las secciones 1.2, 1.3, 2.3, 4.2, 4.3, 4.4, 4.6

[Black, 2001] Black, R. (2001) Managing the Testing Process (2nd edition), John Wiley & Sons: Nueva York.
Véanse las secciones 1. 1, 1. 2, 1.4, 1.5, 2.3, 2.4, 5.1, 5.2, 5.3, 5.5, 5.6

[Buwalda, 2001] Buwalda, H. et al. (2001) Integrated Test Design and Automation, Addison Wesley: Reading, MA
Véase la sección 6.2

[Copeland, 2004] Copeland, L. (2004) A Practitioner's Guide to Software Test Design, Artech House: Norwood, MA
Véanse las secciones 2.2, 2.3, 4.2, 4.3, 4.4, 4.6

[Craig, 2002] Craig, Rick D. and Jaskiel, Stefan P. (2002) Systematic Software Testing, Artech House: Norwood, MA
Véanse las secciones 1.4.5, 2.1.3, 2.4, 4.1, 5.2.5, 5.3, 5.4

[Fewster, 1999] Fewster, M. and Graham, D. (1999) Software Test Automation, Addison Wesley: Reading, MA
Véanse las secciones 6.2, 6.3

[Gilb, 1993]: Gilb, Tom and Graham, Dorothy (1993) Software Inspection, Addison Wesley:

Reading, MA
Véanse las secciones 3.2.2, 3.2.4

[Hetzel, 1988] Hetzel, W. (1988) *Complete Guide to Software Testing*, QED: Wellesley, MA.
Véanse las secciones 1. 3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 4.1, 5.1, 5.3

[Kaner, 2002] Kaner, C., Bach, J. and Pettticord, B. (2002) *Lessons Learned in Software Testing*, John Wiley & Sons: Nueva York.
Véanse las secciones 1.1, 4.5, 5.2

[Myers 1979] Myers, Glenford J. (1979) *The Art of Software Testing*, John Wiley & Sons: Nueva York. Véanse las secciones 1.2, 1.3, 2.2, 4.3

[van Veenendaal, 2004] van Veenendaal, E. (ed.) (2004) *The Testing Practitioner* (Capítulos 6, 8, 10), UTN Publishers: Países Bajos
Véanse las secciones 3.2, 3.3

8. Apéndice A – Antecedentes del programa de estudio

Historia de este documento

Este documento ha sido elaborado entre 2004 y 2007 por un equipo de trabajo compuesto por miembros designados por el Comité Internacional de Cualificación de Pruebas de Software (ISTQB). Inicialmente ha sido revisado por un panel de revisión seleccionado y posteriormente por representantes procedentes de la comunidad internacional de pruebas de software. Las reglas empleadas para la redacción de este documento figuran en el Apéndice C.

Este documento constituye el temario del Certificado Internacional de Nivel Básico en Pruebas de Software, la cualificación internacional de primer nivel aprobada por el ISTQB (www.istqb.org).

Objetivos de la cualificación obtenida mediante el Certificado de Nivel Básico

- Obtener reconocimiento para realizar pruebas en calidad de una especialización de ingeniería de software esencial y profesional
- Facilitar un marco normativo para el desarrollo de las carreras de los probadores
- Permitir que los empleadores, clientes y compañeros otorguen reconocimiento a los probadores cualificados y elevar el perfil de los probadores
- Promover el uso de buenas y coherentes prácticas de pruebas en todas las disciplinas de ingeniería de software
- Identificar los aspectos de las pruebas relevantes y valiosos para el sector
- Ofrecer la oportunidad a los proveedores de software de contratar probadores certificados, obteniendo así una ventaja comercial sobre sus competidores al publicitar su política de selección de probadores.
- Ofrecer la oportunidad a los probadores, y a todos aquellos que estén interesados en el proceso de pruebas, de obtener una cualificación reconocida a nivel internacional en la materia

Objetivos de la Cualificación Internacional (adaptada a raíz de la reunión del ISTQB en Sollentuna, noviembre de 2001)

- Permitir la comparación entre las competencias de prueba en distintos países
- Facilitar el movimiento fronterizo de los probadores
- Hacer que los proyectos multinacionales/internacionales tengan un entendimiento común de los problemas que plantean las pruebas
- Aumentar el número de probadores cualificados a escala mundial
- Tener mayor impacto/valor como una iniciativa a escala internacional en lugar de cómo enfoque específico de un país

- Desarrollar un organismo internacional común de entendimiento y conocimiento sobre pruebas a través del programa y la terminología, y elevar el nivel de conocimiento sobre pruebas de todos los participantes
- Promover la realización de pruebas de software como una profesión en más países
- Dotar a los probadores con una cualificación reconocida en su lengua nativa
- Compartir conocimiento y recursos en todos los países
- Dotar a los probadores con un reconocimiento internacional y esta cualificación gracias a la participación de muchos países

Requisitos de acceso a esta cualificación

Los criterios para acceder al Certificado de Nivel Básico en Pruebas de Software del ISTQB es que los candidatos estén interesados en realizar pruebas de software. No obstante, se recomienda vivamente que los candidatos también:

- Posean unos antecedentes mínimos en el ámbito del desarrollo de software o de las pruebas de software, como por ejemplo seis meses de experiencia como probador de sistema o como probador de pruebas de aceptación del usuario o como desarrollador de software.
- Hayan realizado un curso acreditado conforme a los estándares del ISTQB (por uno de los comités nacionales reconocidos del ISTQB).

Antecedentes e historia del Certificado de Nivel Básico en Pruebas de Software

La certificación independiente de probadores de software nació en el Reino Unido con el Comité Examinador de Sistemas de la Información (ISEB) de la Sociedad Británica de Informática cuando se creó el primer Comité de Pruebas de Software en 1998 (www.bcs.org.uk/iseb). En 2002, ASQF en Alemania empezó a dar soporte a un programa alemán de cualificación de probadores (www.asqf.de). Este programa de estudio está basado en los programas del ISEB y del ASQF; incluye contenido reorganizado, actualizado y adicional, y hace hincapié en temas que serán de gran ayuda práctica a los probadores.

La emisión de un Certificado de Nivel Básico en Pruebas de Software existente (por ejemplo, del ISEBn ASQF o un comité nacional reconocido del ISTQB) anterior a este Certificado Internacional se considerará equivalente al Certificado Internacional. El Certificado de Nivel Básico no caduca y no necesita ser renovado. La fecha en la que ha sido concedido figura en el Certificado.

En cada país participante, los aspectos locales son controlados por un comité nacional de pruebas de software reconocido del ISTQB. El ISTQB determina las obligaciones de los comités nacionales, pero éstas son implementadas dentro de cada país. Las obligaciones de los comités nacionales deben incluir la acreditación de proveedores de formación y la realización de exámenes.

9. Apéndice B - Objetivos de aprendizaje/Nivel de conocimiento cognitivo

Por lo que respecta a este programa de estudio, se han establecido los siguientes objetivos de aprendizaje: Cada tema del programa de estudio será objeto de examen conforme al objetivo de aprendizaje asignado.

Nivel 1: Recordar (K1)

El candidato reconocerá, recordará y retendrá un término o concepto.

Palabras clave: Recordar, recuperar, retener, reconocer, conocer

Ejemplo

Puede reconocer la definición de "fallo" como:

- "La no prestación de servicio a un usuario final o a otra parte interesada" o
- "Una desviación real del componente o sistema de su entrega, servicio o resultado esperado".

Nivel 2: Entender (K2)

El candidato puede seleccionar los motivos o explicaciones de las afirmaciones asociadas al tema, y es capaz de resumir, comparar, clasificar, categorizar y poner ejemplos del concepto de pruebas.

Palabras clave: Resumir, generalizar, abstraer, clasificar, comparar, mapear, contrastar, ejemplificar, interpretar, traducir, representar, inferir, concluir, categorizar, construir modelos

Ejemplos

Puede explicar el motivo por el que las pruebas deben diseñarse lo antes posible:

- Para localizar defectos cuando su resulte más barato eliminarlos.
- Localizar en primer lugar los defectos más importantes.

Puede explicar las similitudes y diferencias entre la integración y las pruebas de sistemas:

- Similitudes: Se prueba más de un componente, y pueden probarse aspectos no funcionales. Diferencias: Las pruebas de integración se centran en interfaces e interacciones y las pruebas de sistema se centran en aspectos globales del sistema, tales como procesos extremo a extremo.

Nivel 3: Aplicar (K3)

El candidato puede seleccionar la aplicación correcta de un concepto o técnica y aplicarla en un contexto dado.

Palabras clave: Implementar, ejecutar, utiliza, seguir un procedimiento, aplicar un

procedimiento

Ejemplo

- Puede identificar valores límite para particiones válidas y no válidas.
- Puede seleccionar casos de prueba a partir de un diagrama dado de transición de estados con vistas a cubrir todas las transiciones.

Nivel 4: Analizar (K4)

El candidato puede clasificar la información sobre un procedimiento o técnica en diferentes partes para su mejor entendimiento, y es capaz de distinguir entre hechos e inferencias. La aplicación típica es analizar un documento, software o situación de proyecto y proponer las acciones pertinentes a adoptar para resolver un problema o tarea.

Palabras clave: Analizar, organizar, encontrar coherencia, integrar, esbozar, estudiar, estructurar, atribuir, deconstruir, diferenciar, discriminar, distinguir, concentrar, seleccionar

Ejemplo

- Analizar riesgos de producto y proponer medidas de mitigación preventivas y correctivas.
- Describir qué partes de un informe de incidencias son factuales y qué partes son inferencias de los resultados.

Bibliografía

(para los niveles cognitivos de los objetivos de aprendizaje)

Anderson, L. W. and Krathwohl, D. R. (eds) (200 1) *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon

10. Apéndice C – Reglas aplicadas al Programa de estudio de Nivel Básico del ISTQB

Programa de estudio de nivel básico

En el desarrollo y la revisión de este programa de estudio se han aplicado las reglas que se enumeran a continuación: (A continuación de cada regla aparece una etiqueta en mayúsculas a modo de abreviatura de la regla)

Reglas generales

SG1. El programa de estudio debe ser entendible y absorbible por personas con 0 a 6 meses (o más) de experiencia en el ámbito de las pruebas de software. (6-MESES)

SG2. El programa de estudio debe ser más práctico que teórico. (PRÁCTICO)

SG3. El programa de estudio debe ser claro y no resultar ambiguo para sus lectores. (CLARO)

SG4. El programa de estudio debe ser entendible para personas de distintos países, y fácilmente traducible a otros idiomas. (TRADUCIBLE)

SG5. El programa de estudio debe emplear inglés americano. (INGLÉS AMERICANO)

Contenido actual

SC1. El programa de estudio debe incluir conceptos de prueba recientes y debe reflejar las mejores prácticas actualmente vigentes en el ámbito de las pruebas de software cuando estén generalmente aceptadas. El programa de estudio puede ser objeto de revisión cada tres o cinco años. (RECIENTE)

SC2. El programa de estudio debe minimizar los problemas asociados al tiempo, tales como las condiciones actuales del mercado, a fin de que pueda tener una vida útil de tres a cinco años. (VIDA ÚTIL)

Objetivos de aprendizaje

LO1. Los objetivos de aprendizaje deben distinguir entre los elementos a reconocer/recordar (nivel cognitivo K1), los elementos que el candidato debe entender a nivel conceptual (K2), los elementos que el candidato debe ser capaz de practicar/utilizar (K3) y los elementos que el candidato debe ser capaz de utilizar para analizar un documento, software o situación de proyecto en un contexto dado (K4) (NIVEL DE CONOCIMIENTO)

LO2. La descripción del contenido debe ser coherente con los objetivos de aprendizaje. (LO-COHERENTE)

LO3. Al objeto de ilustrar los objetivos de aprendizaje, junto con el programa de estudio deben facilitarse ejemplos de preguntas de examen para cada sección principal. (LO-EXAMEN)

Estructura general

LO4. La estructura del programa de estudio debe ser clara y permitir referencias cruzadas a y desde otras partes, a partir de preguntas de examen y a partir de otros documentos relevantes. (REFERENCIAS CRUZADAS)

LO5. Debe minimizarse el solapamiento entre las secciones del programa de estudio. (SOLAPAMIENTO)

LO6. Todas las secciones del programa de estudio deben tener la misma estructura. (ESTRUCTURA CONSISTENTE)

LO7. El programa de estudio debe indicar la versión, la fecha de expedición y el número de página en todas sus páginas. (VERSIÓN)

LO8. El programa de estudio debe incluir una orientación de la cantidad de tiempo a dedicar a cada sección (para reflejar la importancia relativa de cada tema). (TIEMPO DEDICADO)

Referencias

SR1. El programa de estudio facilitará las fuentes y referencias de los conceptos con vistas a ayudar a los proveedores de formación a obtener más información sobre el tema. (REFS)

SR2. Cuando no existan fuentes debidamente identificadas y claras, el programa de estudio deberá aportar más detalles. Así por ejemplo, las definiciones se encuentran en el Glosario, por lo que el programa de estudio se limita a enumerar los términos. (NO DETALLE DE REFERENCIA)

Fuentes de información

Los términos utilizados en el programa de estudio se definen en el Glosario de términos utilizados en pruebas software. Puede acceder a una versión de este glosario en el ISTQB.

En paralelo a este programa de estudio, se publica una lista de lecturas recomendadas sobre pruebas de software. La lista de lecturas principal forma parte de la sección Referencias.

11. Apéndice D – Aviso a los proveedores de formación

A cada título de cada tema principal en este programa de estudio se le ha asignado un tiempo específico en minutos. El objetivo es tanto orientar sobre la proporción de tiempo relativa que debe dedicarse a cada sección en un curso acreditado como establecer un tiempo aproximado a dedicar a la enseñanza de cada sección. Los proveedores de formación pueden dedicar más tiempo del indicado y los candidatos pueden dedicar más tiempo a su lectura e investigación. El currículo de un curso no tiene por qué seguir el mismo orden que el programa de estudio.

El programa de estudio contiene referencias a normas establecidas, que deberán utilizarse durante la preparación de material didáctico Cada norma empleada debe corresponder a la versión indicada en la versión actual de este programa de estudio. Se podrán utilizar y hacer referencia a publicaciones, plantillas o normas que no hayan sido referenciadas en este programa de estudio.

Las áreas específicas del programa de estudio que requieren ejercicios prácticos son las siguientes:

4.3 Técnicas basadas en la especificación o técnicas de caja negra

Aquí deben incluirse trabajos prácticos (ejercicios cortos) que cubran las cuatro técnicas: partición de equivalencia, análisis de valores límite, pruebas de tabla de decisión y pruebas de transición de estado Las lecturas y ejercicios asociados a estas técnicas deben basarse en las referencias previstas para cada técnica.

4.4 Técnicas basadas en la estructura o técnicas de caja blanca

Aquí deben incluirse trabajos prácticos (ejercicios cortos) que permitan evaluar si una serie de pruebas logra o no una cobertura integral de sentencia y decisión, además de diseñar casos de prueba para flujos de control dados.

5.6 Gestión de incidencias

Aquí deben incluirse trabajos prácticos (ejercicios cortos) que cubran la escritura y/o valoración de un informe de incidencias.

12. Apéndice E – Notas de la versión del programa de estudio 2010

1. En el apartado Cambios en los Objetivos de Aprendizaje (LO) se incluyen ciertas aclaraciones.
    a. Se modifica la redacción de los siguientes objetivos de aprendizaje (el contenido y el nivel de los objetivos no varía): LO-1.2.2,LO-1.4.1, LO-2.1.1, LO-2.1.3, LO-4.6.1, LO-6.3.2
    b. Se añade el K4. Motivo: algunos requisitos (LO-4.4.4 y LO-5.6.2) ya se han redactado según el K4, y las preguntas LO-4.6.1 son más sencillas de escribir y examinar en el nivel K4.
    c. LO-1.1.5 ha sido redactado de nuevo y actualizado para K2. Porque cabe esperar una comparación terminológica de términos asociados a los defectos.
    d. LO-1.2.3 Explicar la diferencia entre la actividad de depuración y la de pruebas constituye un nuevo objetivo de aprendizaje. El contenido ya estaba cubierto.
    e. LO-3.1.3 Aspectos de comparación cubiertos.
    f. Se elimina el LO-3.1.4. Era parcialmente redundante con el LO-3.1.3.
    g. Coherencia del LO-3.2.1 con el contenido.
    h. Se actualiza el LO-3.3.2 para K2 para que sea coherente con el LO-3.1.2.
    i. Se elimina el LO-6.1.2 ya que forma parte del LO-6.1.3, que ha sido redactado de nuevo debido al uso inadecuado de una palabra clave K2.
2. Uso coherente del enfoque de pruebas de conformidad con la definición prevista en el glosario. El término estrategia de prueba no será un término a retener.
3. El Capítulo 1.4 ahora incluye el concepto de trazabilidad entre la base de prueba y los casos de prueba.
4. El Capítulo 2.x ahora incluye los objetos de prueba y la base de prueba.
5. El término repetición de pruebas pasa a ser el término principal al igual que en el glosario en lugar de pruebas de confirmación.
6. Se ha añadido el aspecto de la calidad de datos y pruebas en varios puntos del programa de estudio: Calidad de datos y riesgos en los Capítulos 2.2, 5.5, 6.1.8
7. El Capítulo 5.2.3 Criterios de entrada se añade como un nuevo subcapítulo. Motivo: Consistencia con los criterios de salida (-> los criterios de entrada se añaden a LO-5.2.9)
8. Uso consistente de los términos estrategia de pruebas y enfoque de pruebas con la definición prevista en el glosario.
9. Se acorta el capítulo 6.1 porque las descripciones de las herramientas eran

demasiado largas para una clase de 45 minutos.

10. Se ha publicado la norma IEEE 829:2008. Esta versión del programa de estudio no tiene en cuenta esta nueva edición. La sección 5.2 hace referencia al documento “plan maestro de pruebas”. El contenido del plan maestro de pruebas está cubierto ya que se considera que el documento "Plan de prueba" cubre distintos niveles de planificación: pueden crearse planes de prueba para los distintos niveles de prueba, además de un plan de pruebas a nivel del proyecto para cubrir varios niveles de pruebas. Esto último recibe el nombre de “plan maestro de pruebas” en este programa de estudio y en el glosario del ISTQB.
11. El código deontológico ha pasado de ser el CTAL a CTFL.

13. Índice terminológico