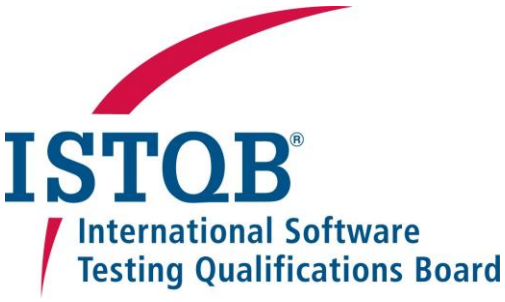
Programa de Estudios de

Probador Certificado de

Nivel Básico

Versión de 2018

Junta Internacional de Calificaciones de Pruebas de Software



Historial de Revisiones

Versión	Fecha	Observaciones
ISTQB 2018	27-abril-2018	Versión de entrega general del candidato
ISTQB 2018	12-febrero-2018	Versión beta del candidato
ISTQB 2018	19-enero-2018	Revisión interna versión 3.0.
ISTQB 2018	15-enero-2018	Revisión interna previa a la versión 2.9, incorporando las ediciones del Equipo Básico.
ISTQB 2018	9-diciembre-2017	Versión Alpha Revisión 2.5 – Edición técnica de la versión 2.0, sin contenido nuevo agregado
ISTQB 2018	22-noviembre-2017	Entrega de la versión Alpha Revisión 2.0 – Programa de Estudios de Probador Certificado de Nivel Básico Actualización Importante de 2018 –vea el Apéndice C - Notas de la Entrega para más detalles
ISTQB 2018	12-junio-2017	Entrega de la versión Alpha – Programa de Estudios de Probador Certificado de Nivel Básico Actualización Importante de 2018 – vea el Apéndice C - Notas de la Entrega
ISTQB 2011	1-abril-2011	Programa de Estudios de Probador Certificado de Nivel Básico Entrega de Mantenimiento, vea Notas de la Entrega
ISTQB 2010	30-marzo-2010	Programa de Estudios de Probador Certificado de Nivel Básico Entrega de Mantenimiento - vea Notas de la Entrega
ISTQB 2007	01-mayo-2007	Programa de Estudios de Probador Certificado de Nivel Básico Entrega de Mantenimiento
ISTQB 2005	01-julio-2005	Programa de Estudios de Probador Certificado de Nivel Básico
ASQF V2.2	julio-2003	ASQF Programa de Estudios Nivel Básico Versión 2.2 “Lehrplan Grundlagen des Software-testens“
ISEB V2.0	25-febrero-1999	ISEB Pruebas de Software Probador Programa de Estudios Básico V2.0

Tabla de Contenidos

Agradecimientos

Este documento fue publicado formalmente por la Asamblea General de la ISTQB (fecha) **Se completará una vez aprobado**.

Fue producido por un equipo de la Junta Internacional de Calificaciones de Pruebas de Software: Klaus Olsen (presidente), Tauhida Parveen (vice-presidente), Rex Black (gerente de proyecto), Debra Friedenberg, Judy McKay, Meile Posthuma, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh y Eshraka Zakaria.

El equipo le agradece a Rex Black y Dorothy Graham por su edición técnica, y al equipo de revisión, al equipo de revisión cruzada y a las Juntas de Miembros por sus sugerencias y aportes.

Las siguientes personas participaron en la revisión, comentarios y votación de este Programa de Estudio: Tom Adams, Tobias Ahlgren, Xu Aiguo, Chris Van Bael, Katalin Balla, Graham Bath, Gualtiero Bazzana, Arne Becher, Veronica Belcher, Lars Hilmar Bjørstrup, Ralf Bongard, Armin Born, Robert Bornelind, Mette Bruhn-Pedersen, Geza Bujdoso, Earl Burba, Filipe Carlos, Young Jae Choi, Greg Collina, Alessandro Collino, Cui Zhe, Taz Daughtrey, Matthias Daigl, Wim Decoutere, Frans Dijkman, Klaudia Dussa-Zieger, Yonit Elbaz, Ofer Feldman, Mark Fewster, Florian Fieber, David Frei, Debra Friedenberg, Conrad Fujimoto, Pooja Gautam, Thorsten Geiselhart, Chen Geng, Christian Alexander Graf, Dorothy Graham, Michel Grandjean, Richard Green, Attila Gyuri, Jon Hagar, Kobi Halperin, Matthias Hamburg, Zsolt Hargitai, Satoshi Hasegawa, Berit Hatten, Wang Hongwei, Tamás Horváth, Leanne Howard, Chinthaka Indikadahena, J. Jayapradeep, Kari Kakkonen, Kakkonen, Gábor Kapros, Beata Karpinska, Karl Kemminger, Sekáu, Seku Kim, Cecilia Kjellman, Johan Klintin, Corne Kruger, Gerard Kruijff, Peter Kunit, Hyeyong Kwon, Bruno Legeard, Thomas Letzkus, Alon Linetzki, Balder Lingegård, Tilo Linz, Hongbiao Liu, Claire Lohr, Ine Lutterman, Marek Majestik , Romanos Matthaios, Judy McKay, Fergus McLachlan, Dénes Medzihradszky, Stefan Merkel, Armin Metzger, Don Mills, Gary Mogyorodi, Ninna Morin, Ingenie Nordström, Adam Novak, Avi Ofer, Magnus C Ohlsson, Joel Oliviera, Monika Stockson, Carolina del Norte, Israel. Francisca Cano Ortiz, Gitte Ottosen, Tuula Pääkkönen, Ana Paiva, Tal Pe'er, Helmut Pichler, Michaël Pilaeten, Horst Pohlmann, Andrew Pollner, Meile Posthuma, Vitalijs Puiso, Salvatore Reale, Stuart Reid, Ralf Reissing, Shark Ren, Mirosos Renda, Randy Rice, Adam Roman, Jan Sabak, Hans Schaefer, Ina Schieferdecker, Franz Schiller, Jianxiong Shen, Klaus Skafte, Mike Smith, Cristina Sobrero, Marco Sogliani, Murian Song, Emilio Soresi, Helder Sousa, Michael Sowers, Michael Stahl, Lucjan Stapp, Li Suyuan, Toby Thompson, Steve Toms, Sagi Traybel, Sabine Uhde, Stephanie Ulrich, Philippos Vakalakis, Erik van Veenendaal, Marianne Vesterdal, Ernst von Düring, Salinda Wickramasinghe, Marie Walsh, Søren Wassard, Hans Weiberg, Paul Weyrosy Young, Surong Yuan, Ester Zabar y Karolina Zmitrowicz.

Grupo de Trabajo de Nivel Básico de la Junta Internacional de Calificaciones de Pruebas de Software (Edición 2018): Klaus Olsen (presidente), Tauhida Parveen (vicepresidente), Rex Black (director de proyecto), Dani Almog, Debra Friedenberg, Rashed Karim, Johan Klintin, Vipul Kocher, Corne Kruger, Sunny Kwon, Judy McKay, Thomas Müller, Igal Levi, Ebbe Munk, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh, Eshraka Zakaria, y Stevan Zivanovic. El equipo central agradece al equipo de revisión y a todas las Juntas de Miembros por sus sugerencias.

Grupo de Trabajo de Nivel Básico de la Junta Internacional de Calificaciones de Pruebas de Software (Edición 2011): Thomas Müller (presidente), Debra Friedenberg. El equipo central agradece al equipo de revisión (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquier Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal), y a todas las Juntas de Miembros por sus sugerencias.

Grupo de Trabajo de Nivel Básico de la Junta Internacional de Calificaciones de Pruebas de Software (Edición 2010): Thomas Müller (presidente) Rahul Verma, Martin Klonk y Armin Beer. El equipo central agradece al equipo de revisión (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Judy McKay, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veenendaal), y a todas las Juntas de Miembros por sus sugerencias.

Grupo de Trabajo de Nivel Básico de la Junta Internacional de Calificaciones de Pruebas de Software (Edición 2007): Thomas Müller (presidente), Dorothy Graham, Debra Friedenberg y Erik van Veenendaal. El equipo central agradece al equipo de revisión (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson, y Wonil Kwon) y a todas las Juntas de Miembros por sus sugerencias.

Grupo de Trabajo de Nivel Básico de la Junta Internacional de Calificaciones de Pruebas de Software (Edición 2005): Thomas Müller (presidente), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson y Erik van Veenendaal. El equipo central agradece al equipo de revisión y a todas las Juntas de Miembros por sus sugerencias.

0 Introducción

0.1 Propósito de este Programa de Estudio

Este Programa de estudios constituye la base para la Calificación Internacional de Pruebas de Software en el Nivel de Básico. La ISTQB proporciona este Programa de estudio de la manera siguiente:

1. A las juntas de los miembros, para que lo traduzcan a su idioma local y acrediten a los proveedores de capacitación. Las Juntas de Miembros pueden adaptar el Programa de estudios a sus necesidades lingüísticas particulares y agregar referencias para adaptarse a sus publicaciones locales.
2. A los organismos de certificación, para derivar preguntas de examen en su idioma local adaptadas a los objetivos de aprendizaje de este Programa de estudio.
3. Para capacitar a los proveedores, para que produzcan cursos y determinen los métodos de enseñanza apropiados.
4. Para los candidatos a la certificación, para que se preparen para el examen de certificación (ya sea como parte de un curso de capacitación o de forma independiente).
5. A la comunidad internacional de software e ingeniería de sistemas, para que progrese la profesión de pruebas de software y de sistemas, y como base para libros y artículos.

La ISTQB puede permitir que otras entidades usen este Programa de estudios para otros propósitos, siempre que soliciten y obtengan un permiso previo por escrito de la ISTQB.

0.2 El Probador Certificado de Nivel Básico en las Pruebas de Software

La calificación de Nivel Básico está dirigida a cualquier persona involucrada en las pruebas de software. Esto incluye personas en roles tales como evaluadores, analistas de prueba, ingenieros de prueba, consultores de prueba, directores de prueba, probadores de aceptación de usuarios y desarrolladores de software. Esta calificación de Nivel Básico también es adecuada para cualquier persona que desee una comprensión básica de las pruebas de software, como son los propietarios de productos, directores de proyecto, gerentes de calidad, gerentes de desarrollo de software, analistas de empresas, directores de TI y consultores de administración. Los titulares del Certificado de Básico podrán continuar con las calificaciones de prueba de software de nivel superior.

La ISTQB Foundation Level Overview 2018 - *Visión General de 2018 del Nivel Básico de la ISTQB de 2018* es un documento separado que incluye la siguiente información:

- Resultados comerciales para el Programa de estudio
- Matriz que muestra la trazabilidad entre los resultados comerciales y los objetivos de aprendizaje
- Resumen de este Programa de Estudio

0.3 Objetivos de aprendizaje examinables y niveles de conocimiento

Los objetivos de aprendizaje apoyan los resultados comerciales y y se utilizan para crear los exámenes de Probador Certificado de Nivel Básico.

En general, todos los contenidos de este programa de estudios son examinables en un nivel K1, a excepción de la Introducción y los Apéndices. Es decir, se le puede pedir al candidato que reconozca, recuerde o memorice una palabra clave o concepto mencionado en cualquiera de los seis capítulos. Los niveles de conocimiento de los objetivos de aprendizaje específicos se muestran al comienzo de cada capítulo y se clasifican de la siguiente manera:

- K1: recordar
- K2: comprender
- K3: aplicar

En el Apéndice B se brindan más detalles y ejemplos de los objetivos de aprendizaje.

Las definiciones de todos los términos enumerados como palabras clave justo debajo de los títulos de los capítulos se recordarán (K1), incluso si no se mencionan explícitamente en los objetivos de aprendizaje.

0.4 El Examen de Certificación de Nivel Básico

El Examen de Certificación de Nivel Básico se basará en este Programa de Estudio. Las respuestas a las preguntas del examen pueden requerir el uso de material basado en más de una sección de este Programa de Estudio. Se pueden examinar todas las secciones del programa de estudios a excepción de la Introducción y los Apéndices. Las normas, los libros y los programas de estudio de la ISTQB pueden incluirse como referencias, pero su contenido no es examinable, más allá de lo que se resume en este plan de estudios de dichos estándares, libros y programas de estudio de la ISTQB.

El formato del examen es de selección múltiple. Hay 40 preguntas. Para aprobar el examen, se debe responder correctamente al menos el 65% de las preguntas (es decir, 26 preguntas).

Los exámenes pueden tomarse como parte de un curso de capacitación acreditado o tomarse de forma independiente (p. ej., en un centro de exámenes o en un examen público). La finalización de un curso de capacitación acreditado no es un requisito previo para el examen.

0.5 Acreditación

Una Junta de Miembros de la ISTQB puede acreditar a proveedores de capacitación cuyo material de curso sigua este Programa de estudio. Los proveedores de capacitación deben obtener las pautas de acreditación de la Junta de Miembros o del organismo que realiza la acreditación. Un curso acreditado se reconoce como que cumple con este programa de estudios y se le permite tener un examen de la ISTQB como parte del curso.

0.6 Nivel de detalle

El nivel de detalle en este programa de estudios permite exámenes y cursos coherentes internacionalmente. Para lograr este objetivo, el programa de estudios consiste en:

- Objetivos generales de instrucción que describen la intención del Nivel Básico
- Una lista de términos que los estudiantes deben poder recordar
- Objetivos de aprendizaje para cada área de conocimiento, que describan el resultado del aprendizaje cognitivo que se logrará
- Una descripción de los conceptos clave, incluidas referencias a fuentes tales como literatura o normas aceptadas

El contenido del programa de estudios no es una descripción de toda el área de conocimiento de las pruebas de software; refleja el nivel de detalle que se cubrirá en los cursos de capacitación de Nivel Básico. Se centra en conceptos y técnicas de prueba que pueden aplicarse a todos los proyectos de software, incluidos los proyectos Ágiles. Este programa de estudios no contiene objetivo de aprendizaje específico alguno relacionado con un ciclo de vida o método de desarrollo de software en particular, pero sí analiza cómo se aplican estos conceptos en proyectos Ágiles, otros tipos de ciclos de vida iterativos e incrementales, y en ciclos de vida secuenciales.

0.7 Cómo está organizado este programa de estudios

Hay seis capítulos con contenido examinable. El encabezado de nivel superior para cada capítulo especifica el tiempo para el capítulo; el tiempo no se proporciona debajo del nivel del capítulo. Para curso de capacitación acreditados, el programa de estudios requiere un mínimo de 16.75 horas de instrucción, distribuidas en los seis capítulos de la siguiente manera:

- Capítulo 1: 175 minutos Fundamentos del proceso de pruebas
- Capítulo 2: 100 minutos Pruebas durante todo el ciclo de vida del software
- Capítulo 3: 135 minutos Las Pruebas Estáticas
- Capítulo 4: 330 minutos Técnicas de prueba
- Capítulo 5: 225 minutos Gestión de prueba
- Capítulo 6: 40 minutos Soporte de herramientas para pruebas

1 Fundamentos del proceso de pruebas — 175 minutos

Palabras clave

cobertura, depuración, defecto, error, falla, calidad, aseguramiento de la calidad, causa raíz, proceso de análisis, base de prueba, caso de prueba, compleción de la prueba, condición de prueba, control de prueba, datos de prueba, diseño de prueba, ejecución de prueba, programa de ejecución de prueba, implementación de prueba, monitorización de la prueba, objeto de prueba, objetivo de prueba, oráculo de prueba, planificación de prueba, procedimiento de prueba, juego de prueba, las pruebas, productos de prueba, trazabilidad, validación, verificación

Objetivos de aprendizaje para los fundamentos de las pruebas:

1.1 ¿Por qué son necesarias las pruebas?

FL-1.1.1 (K1) Identificar los objetivos típicos de las pruebas

FL-1.1.2 (K2) Diferenciar las pruebas de la depuración

1.2 ¿Por qué son necesarias las pruebas?

FL-1.2.1 (K2) Dar ejemplos de por qué es necesario realizar pruebas

FL-1.2.2 (K2) Describir la relación entre las pruebas y la garantía de calidad y dar ejemplos de cómo las pruebas contribuyen a una mayor calidad

FL-1.2.3 (K2) Distinguir entre error, defecto y fallo

FL-1.2.4 (K2) Distinguir entre la causa raíz de un defecto y sus efectos

1.3 Siete principios de la prueba

FL-1.3.1 (K2) Explicar los siete principios de prueba

1.4 Proceso de Prueba

FL-1.4.1 (K2) Explicar el impacto del contexto en el proceso de prueba

FL-1.4.2 (K2) Describir las actividades de prueba y las tareas respectivas dentro del proceso de prueba

FL-1.4.3 (K2) Diferenciar los productos de trabajo que apoyan el proceso de prueba

FL-1.4.4 (K2) Explicar el valor de mantener la trazabilidad entre la base de prueba y los productos de trabajo de prueba

1.5 La psicología de las pruebas

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

FL-1.5.2 (K2) Explicar la diferencia entre el modo de pensar requerido para las actividades de prueba y el modo de pensar requerido para las actividades de desarrollo

1.1 ¿Qué es poner a prueba?

Los sistemas de software son una parte integral de la vida, desde aplicaciones comerciales (p. ej., actividades bancarias) hasta productos de consumo (p. ej., automóviles). La mayoría de las personas han tenido una experiencia con software que no funcionó como se esperaba. El software que no funciona correctamente puede ocasionar muchos problemas, incluida la pérdida de dinero, tiempo o reputación comercial, e incluso lesiones o la muerte. La prueba de software es una forma de evaluar la calidad del software y reducir el riesgo de fallo del software en la operación.

Una percepción errónea común de las pruebas es que solo consiste en ejecutar pruebas, es decir, ejecutar el software y verificar los resultados. Como se describe en la sección 1.4, las pruebas de software son un proceso que incluye muchas actividades diferentes; la ejecución de pruebas (incluida la verificación de resultados) es solo una de estas actividades. El proceso de prueba también incluye actividades como la planificación de prueba, el análisis, el diseño y la implementación de prueba, información del avance y los resultados de las pruebas y la evaluación de la calidad de un objeto de prueba.

Algunas pruebas implican la ejecución del componente o sistema que se está probando; tales pruebas se llaman pruebas dinámicas. Otras pruebas no implican la ejecución del componente o sistema que se está probando; tales pruebas se llaman pruebas estáticas. Por lo tanto, las pruebas también incluyen la revisión de productos de trabajo, tales como requisitos, historias de usuario y código fuente.

Otra percepción errónea de las pruebas es que se enfoca por completo en la verificación de requisitos, historias de usuario u otras especificaciones. Si bien las pruebas implican verificar si el sistema cumple con los requisitos especificados, también implica validación, que es verificar si el sistema cumplirá con las necesidades de los usuarios y otras partes interesadas en sus entornos operativos.

Las actividades de prueba se organizan y se llevan a cabo de manera diferente en diferentes ciclos de vida (ver sección 2.1).

1.1.1 Objetivos típicos de las pruebas

Para cualquier proyecto dado, los objetivos de prueba pueden incluir:

- Evaluar productos de trabajo, tales como requisitos, historias de usuario y código fuente
- Verificar si se cumplen todos los requisitos especificados
- Validar si el objeto de prueba está completo y funciona como esperan los usuarios y otras partes interesadas
- Fomentar la confianza en el nivel de calidad del objeto de prueba
- Prevenir defectos
- Encontrar fallos y defectos
- Proporcionar información suficiente a las partes interesadas que les permitan tomar decisiones informadas, especialmente con respecto al nivel de calidad del objeto de prueba
- Reducir el nivel de riesgo de calidad inadecuada del software (p. ej., fallos no detectadas previamente que ocurren en la operación)
- Cumplir con los requisitos o normas contractuales, legales o reglamentarios, y/o verificar el cumplimiento del objeto de prueba con dichos requisitos o normas

Los objetivos de prueba pueden variar, dependiendo del contexto del componente o sistema que se esté probando, el nivel de prueba y el modelo de ciclo de vida de desarrollo del software. Estas diferencias pueden incluir, p. ej.:

- Durante las pruebas de componentes, un objetivo puede ser encontrar la mayor cantidad de fallos posibles para que los defectos subyacentes se identifiquen y solucionen de manera temprana.
- Otro objetivo puede ser aumentar la cobertura del código de prueba de componentes.
- Durante las pruebas de aceptación, un objetivo puede ser confirmar que el sistema funciona como se espera y cumple con los requisitos.
- Otro objetivo de esta prueba puede ser proporcionar información a las partes interesadas sobre el riesgo de entregar el sistema en un momento dado.

1.1.2 Las pruebas y la depuración

Las pruebas y la depuración son diferentes. La ejecución de pruebas puede mostrar fallas causadas por defectos en el software. La depuración es la actividad de desarrollo que encuentra, analiza y corrige dichos defectos. Las pruebas de confirmación posteriores verifican si los arreglos solucionaron los defectos. En algunos casos, los probadores son responsables de la prueba inicial y la prueba de confirmación final, mientras que los desarrolladores realizan la depuración y la prueba de componentes asociados. Sin embargo, en el desarrollo Ágil y en algunos otros ciclos de vida, los probadores pueden participar en la depuración y en la prueba de componentes.

La norma ISO (ISO/IEC/IEEE 29119-1) tiene más información sobre los conceptos de prueba de software.

1.2 ¿Por qué son necesarias las pruebas?

Las pruebas rigurosas de los componentes y sistemas, y su documentación asociada, pueden ayudar a reducir el riesgo de que ocurran fallos durante la operación. Cuando los defectos se detectan y se arreglan posteriormente, esto contribuye a la calidad de los componentes o sistemas. Además, puede que ser necesario que las pruebas de software cumplan con requisitos contractuales o legales o normas específicas de la industria.

1.2.1 Aportes de las Pruebas al Éxito

A lo largo de la historia de la computación, es bastante común que el software y los sistemas se entreguen en operación y, debido a la presencia de defectos, causen fallas o no satisfagan las necesidades de las partes interesadas. Sin embargo, el uso de técnicas de prueba apropiadas puede reducir la frecuencia de tales entregas problemáticas, cuando esas técnicas se aplican con el nivel adecuado de experiencia en pruebas, en los niveles de prueba adecuados y en los puntos adecuados en el ciclo de vida del desarrollo de software. Los ejemplos incluyen:

- Tener probadores involucrados en revisiones de requisitos o refinamiento de las historias de usuario podría detectar defectos en estos productos de trabajo. La identificación y eliminación de defectos de requisitos reduce el riesgo de que se desarrolle una funcionalidad incorrecta o no verificable.
- Hacer que los probadores trabajen estrechamente con los diseñadores de sistemas mientras se diseña el sistema puede aumentar la comprensión de cada parte del diseño y como probarlo. Esta mayor comprensión puede reducir el riesgo de defectos de diseño fundamentales y permitir que las pruebas se identifiquen en una etapa temprana.
- Hacer que los probadores trabajen estrechamente con los desarrolladores mientras el código está en desarrollo puede aumentar la comprensión de cada parte del código y cómo probarlo. Esta mayor comprensión puede reducir el riesgo de defectos dentro del código y las pruebas.
- Hacer que los probadores verifiquen y validen el software antes de su entrega puede detectar

fallas que de otro modo podrían haberse omitido y apoyar el proceso de eliminar los defectos que causaron las fallas (es decir, la depuración). Esto aumenta la probabilidad de que el software satisfaga las necesidades de los interesados y cumpla con los requisitos.

Además de estos ejemplos, el logro de los objetivos de prueba definidos (consulte la sección 1.1.1) contribuye al éxito general del desarrollo y mantenimiento del software.

1.2.2 Aseguramiento de la calidad y las pruebas

Mientras que las personas a menudo usan la frase *aseguramiento de la calidad* (o simplemente *AC*) para referirse a las pruebas, el aseguramiento de la calidad y las pruebas no son lo mismo, pero están relacionadas. Un concepto más amplio, la gestión de la calidad, los une. La gestión de la calidad incluye todas las actividades que dirigen y controlan una organización con respecto a la calidad.
Entre otras actividades, la gestión de la calidad incluye tanto el aseguramiento de la calidad como el control de la calidad. El aseguramiento de la calidad generalmente se enfoca en la observancia de los procesos adecuados, a fin de brindar confianza en que se lograrán los niveles adecuados de calidad. Cuando los procesos se llevan a cabo correctamente, los productos de trabajo creados por esos procesos son generalmente de mayor calidad, lo que contribuye a la prevención de defectos. Además, el uso del análisis de la causa raíz para detectar y eliminar las causas de los defectos, junto con la aplicación adecuada de los hallazgos de las reuniones retrospectivas para mejorar los procesos, es importante para un control de calidad efectivo.

El control de calidad involucra varias actividades, incluyendo actividades de prueba, que apoyan el logro de niveles adecuados de calidad. Las actividades de prueba son parte del proceso general de desarrollo o mantenimiento de software. Dado que la garantía de calidad se refiere a la ejecución correcta de todo el proceso, la garantía de calidad apoya las pruebas adecuadas. Como se describe en las secciones 1.1.1 y 1.2.1, las pruebas contribuyen al logro de la calidad de diversas maneras.

1.2.3 Errores, Defectos y Fallas

Una persona puede cometer un error (desatino), que puede llevar a la introducción de un defecto (falla o bug) en el código del software o en algún otro producto de trabajo relacionad. Un error que conduzca a la introducción de un defecto en un producto de trabajo puede desencadenar un error que conduzca a la introducción de un defecto en un producto de trabajo relacionado. Por ejemplo, un error de obtención de requisitos puede llevar a un defecto de requisitos, lo que a su vez da como resultado un error de programación que conduce a un defecto en el código.

Si se ejecuta un defecto en el código, esto puede causar una falla, pero no necesariamente en todas las circunstancias. Por ejemplo, algunos defectos requieren entradas o condiciones previas muy específicas para desencadenar una falla, que puede ocurrir raramente o nunca.

Los errores pueden ocurrir por muchas razones, tales como:

- La presión de tiempo
- Falibilidad humana
- Participantes en el proyecto sin experiencia o insuficientemente capacitados
- Falta de comunicación entre los participantes del proyecto, incluida la falta de comunicación sobre los requisitos y el diseño
- Complejidad del código, diseño, arquitectura, el problema subyacente a resolver y/o las tecnologías utilizadas
- Malentendidos sobre las interfaces dentro del sistema y entre sistemas, especialmente cuando tales interacciones dentro del sistema y entre sistemas son grandes
- Nuevas tecnologías poco comunes

Además de las fallas causadas por defectos en el código, las fallas también pueden ser causadas por

condiciones ambientales. Por ejemplo, la radiación, los campos electromagnéticos y la contaminación pueden causar defectos en el firmware o influir en la ejecución del software al cambiar las condiciones del hardware.

No todos los resultados inesperados de las pruebas son fallos. Los falsos positivos pueden ocurrir debido a errores en la forma en que se ejecutaron las pruebas, o debido a defectos en los datos de prueba, el entorno de prueba u otro software de prueba, o por otras razones. También puede ocurrir lo inverso, donde errores o defectos similares conducen a falsos negativos. Los falsos negativos son pruebas que no detectan defectos que deberían haber detectado; los falsos positivos se reportan como defectos, pero en realidad no son lo son.

1.2.4 Defectos, Causas raíz y Efectos

Las causas raíz de los defectos son las acciones o condiciones más tempranas que contribuyeron a crear los defectos. Los defectos pueden analizarse para identificar sus causas raíz, a fin de reducir la aparición de defectos similares en el futuro. Al centrarse en las causas principales más importantes, el análisis de las causas principales puede conducir a mejoras en los procesos que impiden la introducción de un número significativo de defectos futuros.

Por ejemplo, supongamos que los pagos incorrectos de intereses, debido a una sola línea de código incorrecto, generen quejas de los clientes. El código defectuoso se escribió para una historia de usuario que era ambigua, debido a la mala interpretación del propietario del producto sobre cómo calcular los intereses. Si existe un gran porcentaje de defectos en los cálculos del interés, y estos defectos tienen su causa raíz en malentendidos similares, los propietarios del producto podrían recibir capacitación sobre el tema de los cálculos de los intereses para reducir dichos defectos en el futuro.

En este ejemplo, las quejas de los clientes son efectos. Los pagos de intereses incorrectos son fallos. El cálculo incorrecto en el código es un defecto, y resultó del defecto original, la ambigüedad en la historia de usuario. La causa raíz del defecto original fue la falta de conocimiento por parte del propietario del producto, lo que dio como resultado que el propietario del producto cometiera un error al escribir la historia de usuario. El proceso de análisis de la causa raíz se discute en la ISTQB-ETM Programa de Estudios de Gestión de Pruebas de Nivel de Expertos y ISTQB-EITP Mejora del Programa de Estudios del Proceso de Prueba de Nivel de Expertos.

1.3 Siete Principios de las Pruebas

Varios principios de pruebas se han sugerido durante los pasados 50 años y estos ofrecen pautas generales comunes para todas las pruebas.

1. Las pruebas muestran la presencia de defectos, no su ausencia

Las pruebas pueden mostrar que los defectos están presentes, pero no pueden probar que no haya defectos. Las pruebas reducen la probabilidad de que queden defectos sin detectar en el software, pero, incluso si no se encuentran defectos, las pruebas no son una prueba de exactitud.

2. Las pruebas exhaustivas son imposibles

Probarlo todo (todas las combinaciones de entradas y condiciones previas) no es factible, excepto en casos triviales. En lugar de intentar realizar pruebas exhaustivas, se debe utilizar el análisis de riesgos, las técnicas de prueba y las prioridades para enfocar los esfuerzos de prueba.

3. Las pruebas tempranas ahorran tiempo y dinero

Para detectar defectos de manera temprana, las actividades de prueba tanto estáticas como dinámicas deben iniciarse lo antes posible en el ciclo de vida del desarrollo de software. Las pruebas tempranas en ocasiones se denominan *desplazamiento a la izquierda*. Las pruebas tempranas en el ciclo de vida de desarrollo del software ayudan a reducir o eliminar cambios costosos (consulte la sección 3.1).

4. Los defectos se agrupan

Una pequeña cantidad de módulos generalmente contiene la mayoría de los defectos detectados durante las pruebas previas a la entrega, o es responsable de la mayoría de los fallos funcionales. Los grupos de defectos pronosticados y los grupos de defectos reales observados en la prueba o la operación constituyen una contribución importante a un análisis de riesgo utilizado para enfocar el esfuerzo de prueba (como se menciona en el principio 2).

5. Cuidado con la paradoja del pesticida

Si las mismas pruebas se repiten una y otra vez, con el tiempo los mismos casos de prueba ya no encontrarán nuevos defectos. Para detectar nuevos defectos, es posible que las pruebas existentes y los datos de las pruebas deban cambiar y que se deba escribir nuevas pruebas. (Las pruebas ya no son eficaces para detectar defectos, al igual que los pesticidas ya no son eficaces para matar insectos después de un tiempo). En algunos casos, como las pruebas de regresión automatizadas, la paradoja del pesticida tiene un resultado beneficioso, el cuál es el número relativamente bajo de defectos de regresión.

6. Las pruebas dependen del contexto

Las pruebas se realizan de manera diferente en diferentes contextos. Por ejemplo, el software de control industrial crítico para la seguridad se prueba de forma diferente a una aplicación móvil de comercio electrónico. Como otro ejemplo, las pruebas en un proyecto Ágil se realizan de manera diferente a las pruebas en un proyecto de ciclo de vida secuencial (consulte la sección 2.1).

7. La ausencia de errores es una falacia

Algunas organizaciones esperan que los probadores puedan realizar todas las pruebas posibles y encontrar todos los defectos posibles, pero los principios 2 y 1, respectivamente, nos dicen que esto es imposible. Además, es una falacia (es decir, una creencia errónea) esperar que *simplemente* encontrar y corregir una gran cantidad de defectos garantice el éxito de un sistema. Por ejemplo, probar a fondo todos los requisitos especificados y corregir todos los defectos detectados aún podría producir un sistema que difícil de utilizar, que no cumpla con las necesidades y expectativas de los usuarios, o que sea inferior en comparación con otros sistemas de la competencia.

Consulte a Myers 2011, Kaner 2002 y Weinberg 2008 para ver ejemplos de estos y otros principios de prueba.

1.4 Proceso de Prueba

No existe un proceso de prueba de software universal, pero existen conjuntos comunes de actividades de prueba sin las cuales la prueba tendrá menos probabilidades de alcanzar sus objetivos establecidos. Estos conjuntos de actividades de prueba constituyen un proceso de prueba. El proceso de prueba de software adecuado y específico en cualquier situación depende de muchos factores.
Las actividades de prueba que están involucradas en este proceso de prueba, la manera en que se implementan estas actividades y cuándo ocurren, pueden discutirse en la estrategia de prueba de una organización.

1.4.1 Proceso de Prueba en Contexto

Los factores contextuales que influyen en el proceso de prueba para una organización incluyen, entre otros, los siguientes:

- Modelo de ciclo de vida de desarrollo de software y metodologías de proyectos que se utilizan
- Los niveles y los tipos de prueba que se tienen en cuenta
- Riesgos del producto y del proyecto
- Dominio empresarial
- Las limitaciones operacionales, que incluyen, pero no se limitan a:
    - Presupuestos y recursos

  - Plazos de entrega
  - Complejidad
  - Requisitos contractuales y regulatorios
- Políticas y prácticas organizativas
- Normas internas y externas requeridas

Las siguientes secciones describen aspectos generales de los procesos organizativos de prueba referidos a lo siguiente:

- Actividades y tareas de prueba
- Prueba de productos de trabajo
- Trazabilidad entre las bases de prueba y los productos de trabajo de prueba

Es muy útil si la base de prueba (para cualquier nivel o tipo de prueba que se esté considerando) tiene definido criterios de cobertura medibles. Los criterios de cobertura pueden actuar eficazmente como indicadores clave de rendimiento (ICR) para impulsar las actividades que demuestren el logro de los objetivos de las pruebas de software (consulte la sección 1.1.1).

Por ejemplo, para una aplicación móvil, la base de prueba puede incluir una lista de requisitos y una lista de dispositivos móviles compatibles. Cada requisito es un elemento de la base de prueba. Cada dispositivo compatible es también un elemento de la base de prueba. Los criterios de cobertura pueden requerir al menos un caso de prueba para cada elemento de la base de prueba. Una vez ejecutados, los resultados de estas pruebas informe a las partes interesadas si se cumplen los requisitos especificados y si se observaron fallos en los dispositivos compatibles.

La norma ISO (ISO/IEC/IEEE 29119-2) tiene más información sobre estos procesos de prueba.

1.4.2 Actividades y tareas de prueba

Un proceso de prueba consiste en los siguientes principales grupos de actividades:

- Planificación de prueba
- Monitorización y control de la prueba
- Análisis de prueba
- Diseño de prueba
- Implementación de prueba
- Ejecución de una prueba
- Compleción de la prueba

Cada grupo de actividades está compuesto por actividades constitutivas, que se describirán en las subsecciones siguientes. Cada actividad dentro de cada grupo de actividades a su vez puede consistir en múltiples tareas individuales, que variarían de un proyecto o entrega a otro.

Además, aunque muchos de estos grupos de actividades pueden parecer lógicamente secuenciales, a menudo se implementan de forma iterativa. Por ejemplo, el desarrollo Ágil implica pequeñas iteraciones de diseño de software, compilación y prueba que se realizan de forma continua, respaldadas por una planificación continua. Por lo tanto, las actividades de prueba también se realizan de forma iterativa y continua dentro de este enfoque de desarrollo. Incluso en el desarrollo secuencial, la secuencia lógica escalonada de actividades implicará superposición, combinación, concurrencia u omisión, por lo que generalmente es necesario adaptar estas actividades principales dentro del contexto del sistema y el proyecto.

Planificación de prueba

La planificación de la prueba implica actividades que definen los objetivos de la prueba y el enfoque para cumplir los objetivos de prueba dentro de las limitaciones impuestas por el contexto (p. ej., especificando técnicas y tareas de prueba adecuadas, y formulando un calendario de prueba para cumplir con una fecha límite). Los planes de prueba pueden revisarse en función de los comentarios de las actividades de monitorización y control. La planificación de la prueba se explica con más detalle en la sección 5.2.

Monitorización y control de la prueba

La monitorización de la prueba implica la comparación continua del progreso real con el plan de prueba utilizando cualquier métrica de monitorización de la prueba definida en el plan de prueba. El control de prueba implica tomar las acciones necesarias para cumplir los objetivos del plan de prueba (que puede actualizarse a medida que pasa el tiempo). La monitorización y el control de las pruebas están respaldados por la evaluación de los criterios de salida, a los que se hace referencia como la definición de «hecho» en algunos ciclos de vida (consulte ISTQB-AT Extensión del Programa de Estudio de Probador Ágil de Nivel Básico). Por ejemplo, la evaluación de los criterios de salida para la ejecución de la prueba como parte de un nivel de prueba dado puede incluir:

- Verificar los resultados de las pruebas y los registros con los criterios de cobertura especificados
- Evaluar el nivel de calidad de los componentes o sistemas en función de los resultados de las pruebas y los registros
- Determinar si se necesitan más pruebas (p. ej., si las pruebas originalmente destinadas a alcanzar un cierto nivel de cobertura de riesgo del producto no lo hicieron, lo que requiere que se escriban y ejecuten pruebas adicionales)

El progreso de la prueba con respecto al plan se comunica a las partes interesadas en los informes de progreso de la prueba, incluidas las desviaciones del plan y la información para respaldar cualquier decisión de detener la prueba.

La monitorización y control de las pruebas se explican con más detalle en la sección 5.3.

Análisis de prueba

Durante el análisis de prueba, la base de la prueba se analiza para identificar características verificables y definir las condiciones de prueba asociadas. En otras palabras, el análisis de prueba determina “qué probar” en términos de criterios de cobertura medibles.

El análisis de prueba incluye las siguientes actividades principales:

- Analizar la base de prueba adecuada para el nivel de prueba que se considera, por ejemplo:
  - Especificaciones de requisitos, tales como requisitos comerciales, requisitos funcionales, requisitos del sistema, historias de usuario, épicas, casos de uso o productos de trabajo similares que especifiquen el componente funcional o no funcional deseado o el comportamiento del sistema
  - Información de diseño e implementación, como diagramas o documentos de arquitectura del sistema o software, especificaciones de diseño, flujos de llamadas, diagramas de modelado (p. ej., UML o diagramas de entidad-relación), especificaciones de interfaz o productos de trabajo similares que especifican la estructura del sistema o componente
  - La implementación del componente o sistema en sí, incluido el código, los metadatos y las consultas de la base de datos y las interfaces
  - Informes de análisis de riesgos, que pueden considerar aspectos funcionales, no funcionales y estructurales del componente o sistema
- Evaluar la base de prueba y los elementos de prueba para identificar defectos de varios tipos, tales como:
  - Ambigüedades
  - Omisiones

- Inconsistencias
- Inexactitudes
- Contradicciones
- Sentencias superfluas

- Identificar las prestaciones y conjuntos de características que se van a probar
- Definir y priorizar las condiciones de prueba para cada función, según el análisis de la base de prueba, y considerar las características funcionales, no funcionales y estructurales, otros factores comerciales y técnicos, y los niveles de riesgo
- Captura de la trazabilidad bidireccional entre cada elemento de la base de prueba y las condiciones de prueba asociadas (ver secciones 1.4.3 y 1.4.4)

La aplicación de técnicas de prueba de caja negra, caja blanca y basadas en la experiencia puede ser útil en el proceso de análisis de prueba (ver capítulo 4) para reducir la probabilidad de omitir condiciones de prueba importantes y definir condiciones de prueba más precisas y exactas.

En algunos casos, el análisis de prueba produce condiciones de prueba que se deben utilizar como objetivos de prueba en los contratos de la prueba. Los contratos de la prueba son productos de trabajo típicos en algunos tipos de pruebas basadas en la experiencia (consulte la sección 4.4.2). Cuando estos objetivos de prueba son trazables a la base de prueba, se puede medir la cobertura lograda durante dichas pruebas basadas en la experiencia.

La identificación de defectos durante el análisis de prueba es un beneficio potencial importante, especialmente cuando no se utiliza otro proceso de revisión y/o el proceso de prueba está estrechamente relacionado con el proceso de revisión. Dichas actividades de análisis de prueba no solo verifican si los requisitos son coherentes, se expresan y completan adecuadamente, sino que además validan si los requisitos capturan adecuadamente las necesidades de los clientes, usuarios y otras partes interesadas. Por ejemplo, técnicas como el desarrollo guiado por el comportamiento (BDD) y el desarrollo guiado por pruebas de aceptación (ATDD), que implican generar condiciones de prueba y casos de prueba a partir de historias de usuario y criterios de aceptación antes de la codificación, también verifican, validan y detectan defectos en las historias de usuario y criterios de aceptación (consulte ISTQB Extensión del Programa de estudios de Probador Ágil de Nivel Básico).

Diseño de prueba

Durante el diseño de la prueba, las condiciones de prueba se detallan en casos de prueba de alto nivel, conjuntos de casos de prueba de alto nivel y otro software de prueba. Entonces, el análisis de prueba responde a la pregunta “¿qué probar?”, Mientras que el diseño de prueba responde a la pregunta “¿Cómo realizar la prueba?”

El diseño de prueba incluye las siguientes actividades principales:

- Diseñar y priorizar casos de prueba y conjuntos de casos de prueba
- Identificar los datos de prueba requeridos para las condiciones de prueba y casos de prueba
- Diseñar el inicio del entorno de prueba e identificar la infraestructura y herramientas necesarias
- Verificar y actualizar la trazabilidad bidireccional entre la base de prueba, las condiciones de prueba, los casos de prueba y los procedimientos de prueba (consulte la sección 1.4.4)

La elaboración de condiciones de prueba en casos de prueba y conjuntos de casos de prueba durante el diseño de prueba a menudo implica el uso de técnicas de prueba (ver capítulo 4).

Al igual que con el análisis de prueba, el diseño de prueba también puede resultar en la identificación de tipos similares de defectos en la base de la prueba. También como sucede con el análisis de prueba, la identificación de defectos durante el diseño de prueba es un beneficio potencial importante.

Implementación de prueba

Implementación de prue Durante la implementación de prueba, se crea y/o completa el software de prueba necesario para la ejecución de la prueba, incluida la secuenciación de los casos de prueba en los procedimientos de prueba ba. Por lo que, el diseño de prueba responde a la pregunta "¿cómo realizar la prueba?", Mientras que la implementación de prueba responde a la pregunta "¿Ahora tenemos todo en su sitio para ejecutar las pruebas?"

La implementación de prueba incluye las siguientes actividades principales:

- Desarrollar y priorizar procedimientos de prueba y, potencialmente, crear guiones de prueba automatizados
- Crear juegos de prueba a partir de los procedimientos de prueba y (si los hay) guiones de prueba automatizados
- Organizar los juegos de prueba dentro de un programa de ejecución de prueba de manera que resulte en una ejecución de prueba eficiente (ver sección 5.2.4)
- Crear el entorno de prueba (incluidos, potencialmente, arneses de prueba, virtualización de servicios, simuladores y otros elementos de infraestructura) y verificar que todo lo necesario se haya configurado correctamente
- Preparar datos de prueba y asegurarse de que se cargan correctamente en el entorno de prueba
- Verificar y actualizar la trazabilidad bidireccional entre la base de prueba, las condiciones de prueba, los casos de prueba, los procedimientos de prueba y los juegos de prueba (consulte la sección 1.4.4)

Las tareas de diseño de prueba e implementación de prueba a menudo se combinan.

En las pruebas exploratorias y otros tipos de pruebas basadas en la experiencia, el diseño y la implementación de pruebas pueden ocurrir, y pueden documentarse, como parte de la ejecución de prueba. Las pruebas exploratorias pueden basarse en contratos de la prueba (producidos como parte del análisis de prueba), y las pruebas exploratorias se ejecutan de forma inmediata a medida que se diseñan e implementan (consulte la sección 4.4.2).

Ejecución de la prueba

Durante la ejecución de la prueba, los juegos de prueba se ejecutan de acuerdo con el programa de ejecución de prueba. La ejecución de la prueba incluye las siguientes actividades principales:

- Registrar las ID y las versiones de los elementos de prueba o del objeto de prueba, las herramientas de prueba y el software de prueba
- Ejecutar pruebas ya sea manualmente o empleando herramientas de ejecución de la prueba
- Comparar los resultados reales con los resultados esperados
- Analizar anomalías para establecer sus posibles causas (p. ej., pueden producirse fallos debido a defectos en el código, pero también pueden producirse falsos positivos [consulte la sección 1.2.3])
- Reportar los defectos basados en los fallos observados (ver sección 5.6)
- Registrar el resultado de la ejecución de prueba (p. ej., pasa, falla, bloqueada)
- Repetir actividades de prueba, ya sea como resultado de la acción tomada por una anomalía, o como parte de la prueba planificada (p. ej., la ejecución de una prueba corregida, prueba de confirmación y/o prueba de regresión)
- Verificar y actualizar la trazabilidad bidireccional entre la base de prueba, las condiciones de prueba, los casos de prueba, los procedimientos de prueba y los resultados de prueba.

Compleción de la prueba

Las actividades de compleción de la prueba recopilan datos de las actividades de prueba completadas para consolidar la experiencia, el software de prueba y cualquier otra información relevante. Las actividades de compleción de la prueba se producen en los hitos del proyecto, como cuando se entrega un sistema de software, se completa (o se cancela) un proyecto de prueba, se termina una iteración de un proyecto Ágil (p. ej., como parte de una reunión retrospectiva), se completa un nivel de prueba, o se ha completado una entrega de mantenimiento.

La compleción de la prueba incluye las siguientes actividades principales:

- Verificar si todos los informes de defectos están cerrados, ingresando solicitudes de cambio o elementos de la cartera de pedidos del producto para detectar cualquier defecto que quede sin resolver al final de la ejecución de prueba
- Crear un informe de resumen de prueba para ser comunicado a las partes interesadas
- Finalizar y archivar el entorno de prueba, los datos de prueba, la infraestructura de prueba y otros programas de prueba para posterior reutilización
- Entregar el software de prueba a los equipos de mantenimiento, a otros equipos de proyectos y/u otras partes interesadas que podrían beneficiarse de su uso
- Analizar las lecciones aprendidas de las actividades de prueba completadas para determinar los cambios necesarios para futuras iteraciones, entregas y proyectos
- Utilizar la información recopilada para mejorar la madurez del proceso de prueba

1.4.3 Prueba de productos de trabajo

La prueba de productos de trabajo de se crea como parte del proceso de prueba. Así como existe una variación significativa en la forma en que las organizaciones implementan el proceso de prueba, también existe una variación significativa en los tipos de productos de trabajo creados durante ese proceso, en la forma en que esos productos de trabajo se organizan y administran, y en los nombres que se usan para los mismos. Este programa de estudio sigue el proceso de prueba descrito anteriormente, y los productos de trabajo descritos en este programa de estudios y el Glosario de la ISTQB. La norma ISO (ISO/IEC/IEEE 29119-3) también puede servir como una guía para la prueba de productos de trabajo.

Muchas de las pruebas de productos de trabajo descritas en esta sección pueden capturarse y administrarse utilizando herramientas de gestión de pruebas y herramientas de gestión de defectos (ver el capítulo 6).

Planificación de prueba de productos de trabajo

La planificación de la prueba de productos de trabajo suele incluir uno o más planes de prueba. El plan de prueba incluye información sobre la base de prueba, a la que se relacionarán las otras pruebas de productos de trabajo a través de la información de trazabilidad (consulte a continuación y la sección 1.4.4), así como los criterios de salida (o definición de hecho) que se utilizarán durante la monitorización y el control de prueba. os planes de prueba se describen en la sección 5.2.

Monitorización y control de la prueba de productos de trabajo

La monitorización y control de la prueba de productos de trabajo por lo general incluye varios tipos de informes de prueba, incluidos los informes del avance de la prueba (producidos de forma continua y/o periódica) e informes resúmenes de prueba (producidos en varios hitos de compleción). Todos los informes de prueba deben proporcionar detalles relevantes para la audiencia sobre el avance de la prueba a partir de la fecha del informe, incluido el resumen de los resultados de la ejecución de prueba una vez que estén disponibles.

La monitorización y control de la prueba de productos de trabajo también debe abordar las preocupaciones de la administración del proyecto, como la compleción de tareas, la asignación y uso de recursos y el esfuerzo.

La monitorización y control de las pruebas, y los productos de trabajo creados durante estas actividades se explican con más detalle en la sección 5.3 del presente Programa de Estudio.

Análisis de prueba de productos de trabajo

El análisis de prueba de productos de trabajo incluye condiciones de prueba definidas y priorizadas, cada una de las cuales es en el mejor de los casos, bidireccionalmente rastreable a los elementos específicos de la base de prueba que cubre. Para pruebas exploratorias, el análisis de prueba puede implicar la creación de contratos de prueba. El análisis de la prueba también puede resultar en la detección y reporte de defectos en la base de prueba.

Diseño de prueba de productos de trabajo

Diseñar y priorizar casos de prueba y conjuntos de casos de prueba para poner en práctica las condiciones de prueba definidas en el análisis de prueba. A menudo es una buena práctica diseñar casos de prueba de alto nivel, sin valores concretos para los datos de entrada y los resultados esperados. Estos casos de prueba de alto nivel son reutilizables a lo largo de múltiples ciclos de prueba con diferentes datos concretos, a la vez que documentan adecuadamente el alcance del caso de prueba. Idealmente, cada caso de prueba es trazable bidireccionalmente a las condiciones de prueba que cubre.

El diseño de la prueba también da como resultado el diseño y/o la identificación de los datos de prueba necesarios, el diseño del entorno de prueba y la identificación de la infraestructura y las herramientas, aunque la medida en que se documentan estos resultados varía significativamente.

Las condiciones de prueba definidas en el análisis de prueba se pueden refinar aún más en el diseño de la prueba.

Implementación de prueba de productos de trabajo

La implementación de prueba de productos de trabajo incluye:

- Los procedimientos de prueba y la secuencia de esos procedimientos de prueba
- Juegos de prueba
- Un cronograma de ejecución de pruebas

En el mejor de los casos, una vez que se completa la implementación de prueba, se puede demostrar el cumplimiento de los criterios de cobertura establecidos en el plan de prueba a través de la trazabilidad bidireccional entre los procedimientos de prueba y los elementos específicos de la base de prueba, a través de los casos de prueba y las condiciones de prueba.

En algunos casos, la implementación de prueba implica la creación de productos de trabajo con herramientas utilizadas, como la virtualización de servicios y los guiones de prueba automatizados.

La implementación de prueba también puede resultar en la creación y verificación de los datos de prueba y el entorno de prueba. La integridad de la documentación de los datos y/o los resultados de la verificación del entorno puede variar significativamente.

Los datos de prueba sirven para asignar valores concretos a las entradas y los resultados esperados de los casos de prueba. Estos valores concretos, junto con instrucciones explícitas sobre el uso de los valores concretos, convierten los casos de prueba de alto nivel en casos de prueba ejecutables de bajo nivel. El mismo caso de prueba de alto nivel puede usar diferentes datos de prueba cuando se ejecuta en diferentes entregas del objeto de prueba. Los resultados esperados concretos que se asocian con datos de prueba concretos se identifican mediante el uso de un oráculo de prueba.

En las pruebas exploratorias, se pueden crear algunos diseños e implementación de pruebas de productos

de trabajo durante la ejecución de la prueba, aunque la medida en que se documentan las pruebas exploratorias (y su trazabilidad a elementos específicos de la base de prueba) puede variar significativamente.

Las condiciones de prueba definidas en el análisis de prueba se pueden refinar aún más en la implementación de prueba.

Ejecución de la prueba de productos de trabajo

La ejecución de la prueba de productos de trabajo incluye:

- Documentación del estado de casos de prueba individuales o procedimientos de prueba (p. ej., listo para ejecutar, pasa, falla, bloqueado, omitido deliberadamente, etc.)
- Informes de defecto (ver sección 5.6)
- Documentación sobre los elementos de prueba, los objetos de prueba, las herramientas de prueba y el software de prueba que participaron en la prueba

En el mejor de los casos, una vez que se completa la ejecución de prueba, el estado de cada elemento de la base de prueba se puede determinar y reportar a través de la trazabilidad bidireccional con los procedimientos de prueba asociados. Por ejemplo, podemos decir qué requisitos han pasado todas las pruebas planificadas, qué requisitos han fallado las pruebas y/o tienen defectos asociados con ellas, y qué requisitos de las pruebas planificadas están aún pendientes de ejecución. Esto permite verificar que se han cumplido los criterios de cobertura y permite informar los resultados de las pruebas en términos que sean comprensibles para las partes interesadas.

e la prueba de los productos de trabajo

La compleción de la prueba de los productos de trabajo incluye informes de resumen de prueba, elementos de acción para la mejora de proyectos o iteraciones subsiguientes (p. ej., siguiendo un proyecto retrospectivo Ágil), solicitudes de cambio o elementos de la cartera de pedidos del producto, y software de prueba finalizado.

1.4.4 Trazabilidad entre las bases de prueba y los productos de trabajo de prueba

Como se mencionó en la sección 1.4.3, la prueba de productos de trabajo y los nombres de esos productos de trabajo varían considerablemente. Independientemente de estas variaciones, para implementar una monitorización y control de prueba efectivos, es importante establecer y mantener la trazabilidad a lo largo del proceso de prueba entre cada elemento de la base de prueba y los diversos productos de trabajo de prueba asociados con ese elemento, como se describió anteriormente. Además de la evaluación de la cobertura de las pruebas, una buena trazabilidad ofrece:

- Analizar el impacto de los cambios
- Hacer que la prueba sea auditable
- Cumplir con los criterios de dirección de TI
- Mejorar la capacidad de ser entendidos los informes del avance de la prueba y los informes resúmenes de prueba para incluir el estado de los elementos de la base de prueba (p. ej., los requisitos que pasaron sus pruebas, los requisitos que no pasaron sus pruebas y los requisitos que tienen pruebas pendientes)
- Describirle a las partes interesadas los aspectos técnicos de las pruebas en términos que puedan entender
- Proporcionar información para evaluar la calidad del producto, la capacidad del proceso y el progreso del proyecto en relación con los objetivos comerciales

Algunas herramientas de gestión de pruebas proporcionan modelos de productos de trabajo de prueba que coinciden con parte o con todos los productos de trabajo de prueba descritos en esta sección. Algunas

organizaciones construyen sus propios sistemas de gestión para organizar los productos de trabajo y proporcionar la trazabilidad de la información que necesitan.

1.5 La Psicología de las Pruebas

El desarrollo de software, incluidas las pruebas de software, involucra a seres humanos. Por lo tanto, la psicología humana tiene efectos importantes sobre las pruebas de software.

1.5.1 La psicología humana y las pruebas

La identificación de defectos durante una prueba estática, como son la revisión de requisitos o una sesión de perfeccionamiento de la historia de usuario, o la identificación de fallos durante la ejecución dinámica de la prueba, puede percibirse como una crítica al producto y a su autor. Un elemento de la psicología humana llamado sesgo de confirmación puede dificultar la aceptación de información que no esté de acuerdo con las creencias actuales. Por ejemplo, como los desarrolladores esperan que su código sea correcto, tienen un sesgo de confirmación que dificulta la aceptación de que el código sea incorrecto. Además del sesgo de confirmación, otros sesgos cognitivos pueden dificultar que las personas comprendan o acepten la información producida por las pruebas. Por otra parte, es un rasgo humano común culpar al portador de malas noticias, y la información producida por las pruebas a menudo contiene malas noticias.

Como resultado de estos factores psicológicos, algunas personas pueden percibir las pruebas como una actividad destructiva, aunque contribuyan en gran medida al avance del proyecto y la calidad del producto (consulte las secciones 1.1 y 1.2). Para tratar de reducir estas percepciones, la información sobre defectos y fallos debe comunicarse de manera constructiva. De esta manera, se pueden reducir las tensiones entre los probadores y los analistas, propietarios de productos, diseñadores y desarrolladores. Esto se aplica durante las pruebas tanto estáticas como dinámicas.

Los probadores y jefes de pruebas necesitan tener buenas habilidades interpersonales para poder comunicar de manera efectiva sobre defectos, fallos, resultados de pruebas, avance de las pruebas y riesgos, y para establecer relaciones positivas con los colegas. Las formas de comunicarse bien incluyen los siguientes ejemplos:

- Comience con colaboración en lugar de batallas. Recuérdele a todos el objetivo común de tener mejores sistemas de calidad.
- Haga hincapié sobre los beneficios de las pruebas. Por ejemplo, para los autores, la información sobre los defectos puede ayudarlos a mejorar sus productos de trabajo y sus habilidades. Para la organización, los defectos encontrados y reparados durante las pruebas ahorrarán tiempo y dinero, y reducirán el riesgo general para la calidad del producto.
- Comunique los resultados de la prueba y otros hallazgos de una manera neutral y centrada en los hechos, sin criticar a la persona que creó el elemento defectuoso. Redacte informes de defectos objetivos y fácticos, y revise los hallazgos.
- Trate de comprender cómo se sienten las otras personas y las razones por las cuales reaccionan negativamente ante la información.
- Confirme que la otra persona ha entendido lo que le ha dicho y viceversa.

Los objetivos típicos de la prueba se discutieron con anterioridad (ver sección 1.1). Definir claramente que el conjunto correcto de objetivos de prueba tiene importantes implicaciones psicológicas. La mayoría de las personas tienden a alinear sus planes y comportamientos con los objetivos establecidos por el equipo, la administración y otras partes interesadas. También es importante que los probadores sigan estos objetivos con un mínimo sesgo personal.

1.5.2 Los modos de pensar de los probadores y desarrolladores

Los desarrolladores y los probadores a menudo piensan de manera diferente. El objetivo principal del desarrollo es diseñar y construir un producto. Como se mencionó anteriormente, los objetivos de las pruebas

incluyen la verificación y validación del producto, la búsqueda de defectos antes de la entrega, etc. Estos son diferentes conjuntos de objetivos que requieren diferentes modos de pensar. Reunir estos modos de pensar ayuda a lograr un mayor nivel de calidad del producto.

Un modo de pensar refleja los supuestos de un individuo y los métodos preferidos para la toma de decisiones y la resolución de problemas. El modo de pensar de un probador debe incluir la curiosidad, el pesimismo profesional, un ojo crítico, atención al detalle y una motivación para una comunicación y relaciones buenas y positivas. La mentalidad de un probador tiende a crecer y madurar a medida que el probador gana experiencia.

El modo de pensar de un desarrollador puede incluir algunos de los elementos del modo de pensar de un probador, pero los desarrolladores exitosos a menudo están más interesados en diseñar y crear soluciones que en contemplar lo que podría estar mal con esas soluciones. Además, el sesgo de confirmación hace que sea difícil encontrar errores en su propio trabajo.

Con el modo de pensar correcto, los desarrolladores pueden probar su propio código. Los diferentes modelos de ciclo de vida de desarrollo de software a menudo tienen diferentes formas de organizar los probadores y las actividades de prueba. Tener algunas de las actividades de prueba realizadas por probadores independientes aumenta la efectividad de la detección de defectos, lo cual es particularmente importante para sistemas grandes, complejos o críticos para la seguridad. Los probadores independientes brindan una perspectiva diferente a la de los autores de productos de trabajo (es decir, analistas de negocios, propietarios de productos, diseñadores y programadores), ya que tienen sesgos cognitivos diferentes de los autores.

2 Las Pruebas Durante Todo el Ciclo de Vida del Software

100 minutos

Palabras clave

prueba de aceptación, pruebas alfa, pruebas beta, Software estándar comercial (del inglés *Commercial off-the-shelf* COTS), prueba de integración de componentes, prueba de componentes, pruebas de confirmación, prueba de aceptación contractual, prueba funcional, análisis del impacto, pruebas de integración, prueba de mantenimiento, pruebas no funcionales, prueba de aceptación operacional, prueba de regresión, prueba de aceptación regulatoria, modelo de desarrollo secuencial, prueba de integración de sistemas, prueba de sistema, base de prueba, entorno de prueba, nivel de prueba, objeto de prueba, objetivo de prueba, tipo de prueba, prueba de aceptación del usuario, pruebas de caja blanca

Objetivos de Aprendizaje para Las Pruebas Durante Todo el Ciclo de Vida del Software

2.1 Modelos de ciclo de vida de desarrollo de software

FL-2.1.1 (K2) Explicar las relaciones entre las actividades de desarrollo de software y las actividades de prueba en el ciclo de vida del desarrollo de software

FL-2.1.2 (K1) Identificar las razones por las cuales los modelos de ciclo de vida de desarrollo de software deben adaptarse al contexto del proyecto y a las características del producto

2.2 Niveles de prueba

FL-2.2.1 (K2) Comparar los diferentes niveles de prueba desde la perspectiva de los objetivos, la base de prueba, los objetos de prueba, los defectos y fallos típicos y los enfoques y responsabilidades

2.3 Tipos de prueba

FL-2.3.1 (K2) Comparar pruebas funcionales, no funcionales y de caja blanca

FL-2.3.2 (K1) Reconocer que las pruebas funcionales, no funcionales y de caja blanca se realizan en cualquier nivel de prueba

FL-2.3.3 (K2) Comparar los propósitos de las pruebas de confirmación y las pruebas de regresión

2.4 Pruebas de mantenimiento

FL-2.4.1 (K2) Resumir los desencadenantes para las pruebas de mantenimiento

FL-2.4.2 (K2) Describir el papel del análisis de impacto sobre las pruebas de mantenimiento

2.1 Modelos de ciclo de vida de desarrollo de software

Un modelo de ciclo de vida de desarrollo de software describe los tipos de actividad que se realizan en cada etapa de un proyecto de desarrollo de software, y cómo las actividades se relacionan entre sí de manera lógica y cronológica. Hay una serie de modelos diferentes de ciclo de vida de desarrollo de software, cada uno de los cuales requiere diferentes enfoques para la prueba.

2.1.1 Desarrollo de software y pruebas de software

Una parte importante del papel de un probador es familiarizarse con los modelos comunes de ciclo de vida de desarrollo de software para que puedan llevarse a cabo las actividades de prueba adecuadas.

En cualquier modelo de ciclo de vida de desarrollo de software, existen varias características de las buenas pruebas:

- Para cada actividad de desarrollo, hay una actividad de prueba correspondiente
- Cada nivel de prueba tiene objetivos de prueba específicos para ese nivel
- El análisis y diseño de la prueba para un nivel de prueba dado comienzan durante la actividad de desarrollo correspondiente
- Los probadores participan en las discusiones para definir y refinar los requisitos y el diseño, y participan en la revisión de productos de trabajo (p. ej., requisitos, diseño, historias de usuario, etc.) tan pronto como estén disponibles los borradores

Independientemente del modelo de ciclo de vida de desarrollo de software que se elija, las actividades de prueba deben comenzar en las primeras etapas del ciclo de vida, siguiendo el principio de pruebas de pruebas temprana.

Este programa clasifica los modelos comunes de ciclo de vida de desarrollo de software de la siguiente manera:

- Modelos de desarrollo secuencial
- Modelos de desarrollo iterativo e incremental

Un modelo de desarrollo secuencial describe el proceso de desarrollo de software como un flujo de actividades lineal y secuencial. Esto significa que cualquier fase en el proceso de desarrollo debe comenzar cuando se complete la fase anterior. En teoría, no existe superposición de fases, pero en la práctica, es beneficioso contar con retroalimentación temprana de la siguiente fase.

En el modelo de cascada, las actividades de desarrollo (p. ej., análisis de requisitos, diseño, codificación, pruebas) se completan una tras otra. En este modelo, las actividades de prueba solo ocurren después de que se hayan completado todas las demás actividades de desarrollo.

A diferencia del modelo Cascada, el modelo V integra el proceso de prueba a lo largo del proceso de desarrollo, implementando el principio de prueba temprana. Además, el modelo V incluye niveles de prueba asociados con cada fase de desarrollo correspondiente, lo que apoya las pruebas tempranas (consulte la sección 2.2 para una discusión de los niveles de prueba). En este modelo, la ejecución de las pruebas asociadas con cada nivel de prueba se lleva a cabo de forma secuencial, pero en algunos casos se produce una superposición.

Los modelos de desarrollo secuencial entregan software que contiene el conjunto completo de prestaciones, pero generalmente requieren meses o años para su entrega a las partes interesadas y usuarios.

El desarrollo incremental implica establecer requisitos, diseñar, construir y probar un sistema en partes, lo que significa que las prestaciones del software aumentan gradualmente. La magnitud de los incrementos de estas prestaciones varía, con algunos métodos que tienen partes más grandes y algunas partes más pequeñas. Los incrementos de la prestación pueden ser tan pequeños como un solo cambio a una pantalla de interfaz de usuario o una nueva opción de consulta.

El desarrollo iterativo ocurre cuando los grupos de prestaciones se especifican, diseñan, construyen y prueban juntos en una serie de ciclos, a menudo de una duración fija. Las iteraciones pueden implicar cambios en las prestaciones desarrolladas en iteraciones anteriores, conjuntamente con cambios en el alcance del proyecto. Cada iteración entrega un software funcional que es un subconjunto cada vez mayor del conjunto general de prestaciones hasta que se entrega el software final o se detiene el desarrollo.

Los ejemplos incluyen:

- Proceso Unificado Racional: Cada iteración tiende a ser relativamente larga (p. ej., de dos a tres meses), y los incrementos de prestaciones son igualmente grandes, como dos o tres grupos de prestaciones relacionadas.
- Scrum: Cada iteración tiende a ser relativamente pequeña (p. ej., horas, días, o unas pocas semanas), y los incrementos de prestaciones son igualmente pequeños, como algunas mejoras y/o dos o tres prestaciones nuevas.
- Kanban: Implementado con o sin iteraciones de longitud fija, que pueden ofrecer una mejora o prestación única al finalizar, o pueden agrupar prestaciones para entregarlas a la vez.
- Espiral (o creación de prototipos): Implica la creación de incrementos experimentales, algunos de los cuales se pueden volver a trabajar o incluso abandonar en trabajos de desarrollo posterior.

Los componentes o sistemas desarrollados utilizando estos métodos a menudo implican la superposición y la iteración de niveles de prueba a lo largo del desarrollo, En el mejor de los casos, cada función se prueba en varios niveles de prueba a medida que avanza hacia la entrega. En algunos casos, los equipos utilizan la entrega continua o el despliegue continuo, los cuales implican una automatización significativa de múltiples niveles de prueba como parte de sus conductos de entrega. Muchos esfuerzos de desarrollo que utilizan estos métodos también incluyen el concepto de equipos de autoorganización, que pueden cambiar la forma en que se organiza el trabajo de prueba, así como la relación entre los probadores y los desarrolladores.

Estos métodos forman un sistema en crecimiento, que puede ser entregado a los usuarios finales en forma de prestación por prestación, en una iteración por iteración, o en una forma de entrega principal más tradicional. Independientemente de si los incrementos de software se entregan a los usuarios finales, las pruebas de regresión son cada vez más importantes a medida que el sistema crece.

A diferencia de los modelos secuenciales, los modelos iterativos e incrementales pueden entregar software utilizable en semanas o incluso días; pero solo pueden entregar el conjunto completo de requisitos del producto durante un período de meses o incluso años.

Para obtener más información sobre las pruebas de software en el contexto del desarrollo Ágil, consulte ISTQB-AT Extensión del Programa de estudios de Probador Ágil de Nivel Básico, Black 2017, Crispin 2008 y Gregory 2015.

2.1.2 Modelos de ciclo de vida de desarrollo de software en contexto

Los modelos de ciclo de vida de desarrollo de software deben seleccionarse y adaptarse al contexto de las características del proyecto y del producto. Se debe seleccionar y adaptar un modelo de ciclo de vida de desarrollo de software adecuado en función del objetivo del proyecto, el tipo de producto que se está desarrollando, las prioridades comerciales (p. ej., el tiempo de comercialización) y los riesgos identificados de productos y proyectos. Por ejemplo, el desarrollo y las pruebas de un sistema administrativo interno menor deben diferir del desarrollo y las pruebas de un sistema crítico para la seguridad, como el sistema de control de frenos de un automóvil. Como otro ejemplo, en algunos casos, los problemas organizativos y culturales pueden inhibir la comunicación entre los miembros del equipo, lo que puede impedir el desarrollo iterativo.

Dependiendo del contexto del proyecto, puede ser necesario combinar o reorganizar los niveles de prueba y/o las actividades de prueba. Por ejemplo, para la integración de un producto de software estándar comercial (COTS) en un sistema más grande, el comprador puede realizar pruebas de interoperabilidad en el nivel de prueba de integración del sistema (p. ej., integración a la infraestructura y otros sistemas) y al nivel de prueba de aceptación (funcional y no funcional, junto con pruebas de aceptación del usuario y pruebas de aceptación

operacional). Consulte la sección 2.2 para un debate sobre los niveles de prueba y la sección 2.3 para un debate sobre los tipos de prueba.

Además, los propios modelos de ciclo de vida de desarrollo de software pueden combinarse. Por ejemplo, se puede usar un modelo V para el desarrollo y prueba de los sistemas de servicios de fondo y sus integraciones, mientras que se puede usar un modelo de desarrollo Ágil para desarrollar y probar la interfaz de usuario (IU) y la funcionalidad. La creación de prototipos se puede usar al principio de un proyecto, con un modelo de desarrollo incremental adoptado una vez que se completa la fase experimental.

Los sistemas de Internet de las cosas (IoT), que constan de muchos objetos diferentes, como dispositivos, productos y servicios, generalmente aplican modelos de ciclo de vida de desarrollo de software separados para cada objeto. Esto presenta un desafío particular para el desarrollo de las versiones del sistema de Internet de las cosas. Además, el ciclo de vida de desarrollo de software de tales objetos pone un mayor énfasis en las últimas fases del ciclo de vida de desarrollo de software después de que se han introducido en el uso operativo (p. ej., las fases de operación, actualización y retiro).

2.2 Niveles de prueba

Los niveles de prueba son grupos de actividades de prueba que se organizan y administran juntas. Cada nivel de prueba es una instancia del proceso de prueba, que consiste en las actividades descritas en la sección 1.4, realizadas en relación con el software en un nivel de desarrollo dado, desde unidades o componentes individuales hasta sistemas completos o, cuando corresponda, sistemas de sistemas. Los niveles de prueba están relacionados con otras actividades dentro del ciclo de vida del desarrollo de software. Los niveles de prueba utilizados en este programa son:

- Pruebas de componente
- Pruebas de integración
- Pruebas de sistema
- Pruebas de aceptación

Los niveles de prueba se caracterizan por los siguientes atributos:

- Objetivos específicos
- Base de prueba, referenciada para derivar casos de prueba
- Objetos de prueba (es decir, lo que se está probando)
- Defectos y fallos típicos
- Enfoques y responsabilidades específicas

Para cada nivel de prueba, se requiere un entorno de prueba adecuado. En las pruebas de aceptación, por ejemplo, un entorno de prueba similar a la producción es ideal, mientras que en las pruebas de componente los desarrolladores suelen utilizar su propio entorno de desarrollo.

2.2.1 Pruebas de componente

Objetivos de las pruebas de componente

Las pruebas de componente (también conocidas como pruebas de unidad o módulo) se enfocan en componentes que se pueden probar por separado. Los objetivos de las pruebas de componente incluyen:

- Reducir el riesgo
- Verificar si los comportamientos funcionales y no funcionales del componente son como se diseñaron y especificaron
- Crear confianza en la calidad del componente
- Encontrar defectos en el componente
- Prevenir que los defectos escapen a niveles de prueba más elevados

En algunos casos, especialmente en los modelos de desarrollo incrementales e iterativos (p. ej., Ágil) donde los cambios de código son continuos, las pruebas de regresión automatizada de componentes desempeñan un papel clave en crear confianza en que los cambios no han roto los componentes existentes.

Las pruebas de componente a menudo se realizan en forma aislada del resto del sistema, según el modelo de ciclo de vida del desarrollo del software y el sistema, que puede requerir objetos simulados, virtualización del servicio, arneses, stubs y controladores. Las pruebas de componentes pueden cubrir la funcionalidad (p. ej., la corrección de los cálculos), las características no funcionales (p. ej., la búsqueda de fugas de memoria) y las propiedades estructurales (p. ej., las pruebas de decisión).

Base de prueba

Los ejemplos de productos de trabajo que se pueden usar como base de prueba para pruebas de componente incluyen:

- Diseño detallado
- Código
- Modelo de datos
- Especificaciones de los componentes

Objetos de prueba

Los objetos de prueba típicos para las pruebas de componente incluyen:

- Componentes, unidades o módulos
- Código y estructuras de datos
- Clases
- Módulos de base de datos

Defectos y fallos típicos

Los ejemplos de defectos y fallos típicos para las pruebas de componente incluyen:

- Funcionalidad incorrecta (p. ej., no como se describe en las especificaciones de diseño)
- Problemas de flujo de datos
- Código y lógica incorrectos

Los defectos se suelen solucionar tan pronto como se detectan, a menudo sin una gestión formal de los defectos. Sin embargo, cuando los desarrolladores informan sobre defectos, esto proporciona información importante para el análisis de la causa raíz y la mejora del proceso.

Enfoques y responsabilidades específicas

Las pruebas de componente generalmente las realiza el desarrollador que escribió el código, pero al menos se requiere acceso al código que se está probando. Los desarrolladores pueden alternar el desarrollo de componentes con encontrar y reparar defectos. Los desarrolladores a menudo escriben y ejecutan pruebas después de haber escrito el código para un componente. Sin embargo, especialmente en el desarrollo Ágil, escribir casos de prueba de componentes automatizados puede preceder a escribir el código de la aplicación.

Por ejemplo, tenga en cuenta el desarrollo guiado por pruebas (TDD). El desarrollo guiado por pruebas es altamente iterativo y se basa en ciclos de desarrollo de casos de prueba automatizados, luego construye e integra pequeños fragmentos de código, luego ejecuta las pruebas de componentes, corrige cualquier problema y refactorización del código. Este proceso continúa hasta que el componente se ha construido completamente y pasan todas las pruebas de componente. El desarrollo guiado por pruebas es un ejemplo de un enfoque de «pruebas primero». Si bien el desarrollo guiado por pruebas se originó en eXtreme Programming (XP), se ha extendido a otras formas de Ágil y también a ciclos de vida secuenciales (consulte ISTQB-AT Extensión del Programa de Estudio - Nivel Básico - Probador Ágil).

2.2.2 Pruebas de integración

Objetivos de las pruebas de integración

Las pruebas de integración se centran en las interacciones entre componentes o sistemas. Los objetivos de las pruebas de integración incluyen:

- Reducir el riesgo
- Verificar si los comportamientos funcionales y no funcionales del componente son como se diseñaron y especificaron
- Crear confianza en la calidad de las interfaces
- Encontrar defectos (que pueden estar en las interfaces o en los componentes o sistemas)
- Prevenir que los defectos escapen a niveles de prueba más elevados

Al igual que con las pruebas de componente, en algunos casos, las pruebas de regresión de integración automatizadas brindan confianza en que los cambios no han roto las interfaces, componentes o sistemas existentes.

Existen dos niveles diferentes de pruebas de integración descritos en este programa, que se pueden llevar a cabo en objetos de prueba de diferentes tamaños de la siguiente manera:

- Las pruebas de integración de componentes se centran en las interacciones e interfaces entre componentes integrados. Las pruebas de integración de componentes se realizan después de las pruebas de componente y, por lo general, se automatizan. En el desarrollo iterativo e incremental, las pruebas de integración de componentes suelen ser parte del proceso de integración continua.
- Las pruebas de integración de componentes se centran en las interacciones e interfaces entre sistemas, paquetes, microservicios. Las pruebas de integración del sistema también pueden cubrir las interacciones y las interfaces proporcionadas por organizaciones externas (p. ej., servicios Web). En este caso, la organización en desarrollo no controla las interfaces externas, lo que puede crear varios desafíos para las pruebas (p. ej., garantizar que se solucionen los defectos de bloqueo de prueba en el código de la organización externa, organizar entornos de prueba, etc.). Las pruebas de integración de sistemas se pueden realizar después de las pruebas del sistema o en paralelo con las

actividades de prueba del sistema en curso (tanto en el desarrollo secuencial como en el desarrollo iterativo e incremental).

Base de prueba

Los ejemplos de productos de trabajo que se pueden usar como base de prueba para las pruebas de integración incluyen:

- Diseño de software y de sistema
- Diagramas de secuencia
- Especificaciones de interfaz y protocolo de comunicación
- Casos de uso
- Arquitectura a nivel de componente o de sistema
- Flujos de trabajo
- Definiciones de interfaz externa

Objetos de prueba

Los objetos de prueba típicos para las pruebas de integración incluyen:

- Subsistemas
- Bases de datos
- Infraestructura
- Interfaces
- APIs
- Microservicios

Defectos y fallos típicos

Los ejemplos de defectos y fallos típicos para las pruebas de integración de componentes incluyen:

- Datos incorrectos, datos faltantes o codificación de datos incorrecta
- Secuenciación o sincronización de llamadas de interfaz incorrecta
- Incongruencias de la interfaz
- Fallos en la comunicación entre componentes
- Fallos de comunicación entre componentes no manejados o manejados incorrectamente
- Suposiciones incorrectas sobre el significado, las unidades o los límites de los datos que se pasan entre los componentes

Los ejemplos de defectos y fallos típicos para las pruebas de integración del sistema incluyen:

- Estructuras de mensaje inconsistentes entre sistemas
- Datos incorrectos, datos faltantes o codificación de datos incorrecta
- Incongruencias de la interfaz
- Fallos en la comunicación entre sistemas
- Fallos de comunicación entre sistemas no manejados o manejados incorrectamente

- Suposiciones incorrectas sobre el significado, las unidades o los límites de los datos que se pasan entre los sistemas
- Incumplimiento de las normas de seguridad obligatorias

Enfoques y responsabilidades específicas

Las pruebas de integración de componente y las pruebas de integración de sistemas deben concentrarse en la integración en sí misma. Por ejemplo, si integra el módulo A con el módulo B, las pruebas deben centrarse en la comunicación entre los módulos, no en la funcionalidad de los módulos individuales, ya que eso debería haberse cubierto durante las pruebas de componente. Si integra el sistema X con el sistema Y, las pruebas deben centrarse en la comunicación entre los sistemas, no en la funcionalidad de los sistemas individuales, ya que eso debería haberse cubierto durante las pruebas de sistema. Son aplicables los tipos de pruebas funcionales, no funcionales y estructurales.

Las pruebas de integración de componente a menudo son responsabilidad de los desarrolladores. Las pruebas de integración de sistemas generalmente son responsabilidad de los probadores. En el mejor de los casos, los probadores que realizan pruebas de integración de sistemas deberían comprender la arquitectura del sistema y deberían haber influido en la planificación de la integración.

Si las pruebas de integración y la estrategia de integración se planifican antes de que se construyan los componentes o sistemas, esos componentes o sistemas se pueden construir en el orden requerido para la prueba más eficiente. Las estrategias de integración sistemática pueden basarse en la arquitectura del sistema (p. ej., de arriba hacia abajo y de abajo hacia arriba), tareas funcionales, secuencias de procesamiento de transacciones o algún otro aspecto del sistema o componentes). Para simplificar el aislamiento de defectos y detectar los defectos de manera temprana, la integración normalmente debería ser incremental (es decir, una pequeña cantidad de componentes o sistemas adicionales a la vez) en lugar de “big bang” (es decir, integrar todos los componentes o sistemas en un solo paso). Un análisis de riesgo de las interfaces más complejas puede ayudar a enfocar las pruebas de integración.

Cuanto mayor sea el alcance de la integración, más difícil será aislar los defectos de un componente o sistema específico, lo que puede generar un mayor riesgo y tiempo adicional para la solución de problemas. Esta es una razón por la que la integración continua, donde el software se integra componente por componente (es decir, integración funcional), se ha convertido en una práctica común. Dicha integración continua a menudo incluye pruebas de regresión automatizadas, idealmente en múltiples niveles de prueba.

2.2.3 Pruebas de sistema

Objetivos de las pruebas de sistema

Las pruebas de sistema se centran en el comportamiento y las capacidades de todo un sistema o producto, a menudo considerando las tareas de extremo a extremo que el sistema puede realizar y los comportamientos no funcionales que muestra al realizar esas tareas. Los objetivos de las pruebas de sistema incluyen:

- Reducir el riesgo
- Verificar si los comportamientos funcionales y no funcionales del sistema son como se diseñaron y especificaron
- Validar que el sistema está completo y funcionará como se espera
- Crear confianza en la calidad del sistema como un todo
- Encontrar defectos
- Prevenir que los defectos escapen a niveles de prueba o producción más elevados

Para ciertos sistemas, verificar la calidad de los datos puede ser un objetivo. Al igual que con las pruebas de componente y las pruebas de integración, en algunos casos, las pruebas de regresión automatizadas del sistema brindan la confianza de que los cambios no han roto las funciones existentes o las capacidades de extremo a extremo. Las pruebas del sistema a menudo producen información que usan las partes interesadas para tomar decisiones de entrega. Las pruebas de sistema también pueden satisfacer requisitos o normas legales o reglamentarios.

El entorno de prueba debería corresponder idealmente con el objetivo final o el entorno de producción

Base de prueba

Los ejemplos de productos de trabajo que se pueden usar como base de prueba para las pruebas de sistema incluyen:

- Especificaciones de requisitos del sistema y software (funcionales y no funcionales)
- Informes de análisis de riesgo
- Casos de uso
- Epopeyas e historias de usuario
- Modelos de comportamiento del sistema
- Diagramas de estado
- Manuales de sistema y de usuario

Objetos de prueba

Los objetos de prueba típicos para las pruebas de sistemas incluyen:

- Aplicaciones
- Sistemas de hardware/software
- Sistemas operativos
- Sistema Sujeto a Prueba (SUT)
- Configuración del sistema y datos de configuración

Defectos y fallos típicos

Los ejemplos de defectos y fallos típicos para las pruebas de sistemas incluyen:

- Cálculos incorrectos
- Comportamiento funcional o no funcional del sistema incorrecto o inesperado
- Control incorrecto y/o flujos de datos dentro del sistema
- No realizar correctamente y completamente las tareas funcionales de extremo a extremo
- Fallo del sistema para funcionar correctamente en el (los) entorno (s) de producción
- Fallo del sistema para funcionar como se describe en los manuales de usuario y de sistema

Enfoques y responsabilidades específicas

Las pruebas de sistema deben centrarse en el comportamiento global de extremo a extremo del sistema en su conjunto, tanto funcional como no funcional. Las pruebas de sistema deben usar las técnicas más apropiadas (ver capítulo 4) para los aspectos del sistema que se van a probar. Por ejemplo, se puede crear una tabla de decisiones para verificar si el comportamiento funcional es como se describe en las reglas comerciales.

Los probadores independientes suelen realizar pruebas de sistema. Los defectos en las especificaciones (p. ej., historias de usuario faltantes, requisitos comerciales declarados incorrectamente, etc.) pueden llevar a una falta de comprensión o desacuerdos sobre el comportamiento esperado del sistema. Tales situaciones pueden causar falsos positivos y falsos negativos, lo que desperdicia el tiempo y reduce la efectividad de la detección de defectos, respectivamente. La participación temprana de los probadores en el refinamiento de historias de usuario o en las actividades de pruebas estáticas, como las revisiones, ayuda a reducir la incidencia de tales situaciones.

2.2.4 Pruebas de aceptación

Objetivos de las pruebas de aceptación

Las pruebas de aceptación, como las pruebas de sistema, generalmente se enfocan en el comportamiento y las capacidades de todo un sistema o producto. Los objetivos de las pruebas de aceptación incluyen:

- Crear confianza en la calidad del sistema como un todo
- Validar que el sistema está completo y funcionará como se espera
- Verificar que los comportamientos funcionales y no funcionales del sistema son los especificados

Las pruebas de aceptación pueden generar información para evaluar la preparación del sistema para la implementación y el uso por parte del cliente (usuario final). Los defectos se pueden detectar durante las pruebas de aceptación, pero detectar defectos a menudo no es un objetivo, y encontrar un número significativo de defectos durante las pruebas de aceptación en algunos casos puede considerarse un riesgo importante para el proyecto. Las pruebas de aceptación también pueden satisfacer requisitos o normas legales o reglamentarias.

Las formas comunes de pruebas de aceptación incluyen lo siguiente:

- Pruebas de aceptación del usuario
- Pruebas de aceptación operativa
- Pruebas de aceptación contractuales y regulatorias
- Pruebas alfa y beta.

Cada una se describe en las siguientes cuatro subsecciones.

Pruebas de aceptación del usuario (PAU)

Las pruebas de aceptación de sistemas por parte de los usuarios generalmente se centran en validar la aptitud para el uso del sistema por parte de los usuarios previstos en un entorno operativo real o simulado. El objetivo principal es crear confianza en que los usuarios pueden usar el sistema para satisfacer sus necesidades, cumplir con los requisitos y realizar procesos de negocios con dificultad, costos y riesgos mínimos.

Pruebas de aceptación operativa (PAO)

Las pruebas de aceptación de sistemas por parte del personal de administración de operaciones o sistemas se realizan generalmente en un entorno (simulado) de producción. Las pruebas se centran en aspectos operativos, y pueden incluir:

- Pruebas de copia de seguridad y restauración
- Instalación, desinstalación y actualización
- Recuperación de desastres
- Gestión de usuarios
- Tareas de mantenimiento

- Carga de datos y tareas de migración
- Comprobaciones de las vulnerabilidades de seguridad
- Pruebas de rendimiento

El objetivo principal de las pruebas de aceptación operativa es generar confianza en que los operadores o administradores del sistema pueden mantener el sistema funcionando correctamente para los usuarios en el entorno operativo, incluso en condiciones excepcionales o difíciles.

Pruebas de aceptación contractuales y regulatorias

Las pruebas de aceptación contractuales se realizan según los criterios de aceptación de un contrato para producir software desarrollado a la medida. Los criterios de aceptación deben definirse cuando las partes acuerdan el contrato. Las pruebas de aceptación contractuales a menudo son realizadas por usuarios o por probadores independientes.

Las pruebas de aceptación regulatorias se realizan contra cualquier regulación que se deba cumplir, como las reglamentaciones gubernamentales, legales o de seguridad. Las pruebas de aceptación regulatorias a menudo las realizan usuarios o probadores independientes, en ocasiones con los resultados siendo presenciados o auditados por agencias reguladoras.

El principal objetivo de las pruebas de aceptación contractuales y regulatorias es generar confianza en el cumplimiento contractual o regulatorio.

Pruebas alfa y beta

Las pruebas alfa y beta suelen ser utilizadas por los desarrolladores de software comercial (COTS) que desean obtener comentarios de usuarios, clientes y operadores potenciales o existentes antes de que el producto de software se sitúe en el mercado. Las pruebas alfa se realizan en el sitio de la organización en desarrollo, no por el equipo de desarrollo, sino por clientes potenciales o existentes, y/u operadores o un equipo de prueba independiente. Las pruebas beta son realizadas por clientes potenciales o existentes, y/u operadores en sus propias ubicaciones. La prueba beta puede realizarse después de la prueba alfa, o puede ocurrir sin que se haya producido ninguna prueba alfa anterior.

Uno de los objetivos de las pruebas alfa y beta es generar confianza entre los clientes potenciales o existentes, y/o los operadores, de que pueden usar el sistema en condiciones normales, cotidianas y en el (los) entorno (s) operativo (s) para lograr sus objetivos con la mínima dificultad, costos, y riesgos. Otro objetivo puede ser la detección de defectos relacionados con las condiciones y el (los) entorno (s) en que se utilizará el sistema, especialmente cuando el equipo de desarrollo dificulta la replicación de esas condiciones y el (los) entorno (s).

Base de prueba

Los ejemplos de productos de trabajo que se pueden usar como base de prueba para cualquier prueba de aceptación incluyen:

- Procesos de negocio
- Requisitos de usuario o empresariales
- Regulaciones, contratos legales y normas
- Casos de uso
- Requisitos del sistema
- Documentación del sistema o del usuario
- Procedimientos de instalación
- Informes de análisis de riesgos

Además, como base de prueba para derivar casos de prueba para pruebas de aceptación operacional, se pueden usar uno o más de los siguientes productos de trabajo:

- Procedimientos de copia de seguridad y restauración
- Procedimientos de recuperación de desastres
- Requisitos no funcionales
- Documentación de las operaciones
- Instrucciones de despliegue e instalación
- Objetivos de rendimiento
- Paquetes de base de datos
- Normas o reglamentos de seguridad

Objetos de prueba típicos

Los objetos de prueba típicos para cualquier forma de las pruebas de aceptación incluyen:

- Sistema sujeto a prueba
- Configuración del sistema y datos de configuración
- Procesos de negocio para un sistema totalmente integrado
- Sistemas de recuperación y sitios activos (para pruebas de continuidad del negocio y recuperación de desastres)
- Procesos operacionales y de mantenimiento
- Formularios
- Informes
- Datos de producción existentes y convertidos

Defectos y fallos típicos

Los ejemplos de defectos típicos para cualquier forma de las pruebas de aceptación incluyen:

- Los flujos de trabajo del sistema no cumplen con los requisitos comerciales o del usuario
- Las reglas comerciales no se implementan correctamente
- El sistema no cumple con los requisitos contractuales o reglamentarios
- Fallos no funcionales tales como vulnerabilidades de seguridad, eficiencia de rendimiento inadecuada en cargas elevadas u operación incorrecta en una plataforma compatible

Enfoques y responsabilidades específicas

Las pruebas de aceptación a menudo son responsabilidad de los clientes, usuarios comerciales, propietarios de productos u operadores de un sistema, y también pueden participar otras partes interesadas.

Las pruebas de aceptación a menudo se consideran el último nivel de prueba en un ciclo de vida de desarrollo secuencial, pero también pueden ocurrir en otros momentos, por ejemplo:

- Las pruebas de aceptación de un producto de software COTS puede ocurrir cuando éste está instalado o integrado
- Las pruebas de aceptación de una nueva mejora funcional pueden ocurrir antes de las pruebas de sistema

En el desarrollo iterativo, los equipos de proyecto pueden emplear varias formas de pruebas de aceptación durante y al final de cada iteración, como las que se centran en verificar una nueva prestación en función de sus criterios de aceptación y las enfocadas en validar que una nueva prestación satisface las necesidades de los usuarios. Además, las pruebas alfa y las pruebas beta pueden ocurrir, ya sea al final de cada iteración, después de finalizar cada iteración, o después de una serie de iteraciones. Las pruebas de aceptación del usuario, las pruebas de aceptación operacional, las pruebas de aceptación regulatoria y las pruebas de aceptación contractual también pueden ocurrir, ya sea al final de cada iteración, después de finalizar cada iteración, o después de una serie de iteraciones.

2.3 Tipos de prueba

Un tipo de prueba es un grupo de actividades de prueba destinadas a probar características específicas de un sistema de software, o una parte de un sistema, basado en objetivos de prueba específicos. Tales objetivos pueden incluir:

- Evaluar las características de calidad funcional, como son la integridad, exactitud y pertinencia
- Evaluar las características de calidad no funcionales, como son la confiabilidad, eficiencia del rendimiento, seguridad, compatibilidad y usabilidad
- Evaluar si la estructura o arquitectura del componente o sistema es correcta, completa y según lo especificado
- Evaluar los efectos de los cambios, como son confirmar que los defectos se han solucionado (pruebas de confirmación) y buscar cambios no intencionados en el comportamiento que resulten de los cambios en el software o el entorno (pruebas de regresión)

2.3.1 Pruebas funcionales

Las pruebas funcionales de un sistema involucran pruebas que evalúan las funciones que debe realizar el sistema. Los requisitos funcionales pueden describirse en productos de trabajo, como son especificaciones de requisitos comerciales, epopeyas, historias de usuario, casos de uso o especificaciones funcionales, o pueden no estar documentados. Las funciones son “lo qué” el sistema debe hacer.

Las pruebas funcionales deben realizarse en todos los niveles de prueba (p. ej., las pruebas para componentes pueden basarse en una especificación de componentes), aunque el enfoque es diferente en cada nivel (consulte la sección 2.2).

Las pruebas funcionales tienen en cuenta el comportamiento del software, por lo que se pueden usar técnicas de caja negra para derivar las condiciones de prueba y los casos de prueba para la funcionalidad del componente o sistema (ver sección 4.2).

La minuciosidad de las pruebas funcionales se puede medir a través de la cobertura funcional. La cobertura funcional es la medida en que se ha ejercido algún tipo de elemento funcional mediante pruebas, y se expresa como un porcentaje del tipo de elemento que se está cubriendo. Por ejemplo, al utilizar la trazabilidad entre las pruebas y los requisitos funcionales, se puede calcular el porcentaje de estos requisitos que se abordan mediante pruebas, lo que podría identificar brechas de cobertura.

El diseño y la ejecución de pruebas funcionales pueden involucrar habilidades o conocimientos especiales, como el conocimiento del problema empresarial en particular que resuelve el software (p. ej., software de modelado geológico para las industrias del petróleo y del gas) o la función particular que cumple el software (p. ej., software de juegos de computadora que proporciona entretenimiento interactivo).

2.3.2 Pruebas no funcionales

Las pruebas no funcionales de un sistema evalúan las características de los sistemas y el software, como son la usabilidad, la eficiencia del rendimiento o la seguridad. Consulte la norma ISO (ISO/IEC 25010) para obtener una clasificación de las características de calidad del producto de software. La prueba no funcional es la prueba de “qué tan bien” se comporta el Sistema.

Contrariamente a las percepciones erróneas comunes, las pruebas no funcionales pueden y, a menudo, deben realizarse en todos los niveles de prueba, y hacerlas tan pronto como sea posible. El descubrimiento tardío de defectos no funcionales puede ser extremadamente peligroso para el éxito de un proyecto.

Se pueden usar técnicas de caja negra (ver sección 4.2) para derivar las condiciones de prueba y los casos de prueba para pruebas no funcionales. Por ejemplo, el análisis del valor límite se puede usar para definir las condiciones de estrés para las pruebas de rendimiento.

La minuciosidad de las pruebas no funcionales se puede medir a través de la cobertura no funcional. La cobertura no funcional es la medida en que se ha ejercido algún tipo de elemento no funcional mediante pruebas, y se expresa como un porcentaje del tipo de elemento que se está cubriendo. Por ejemplo, al usar la trazabilidad entre las pruebas y los dispositivos compatibles para una aplicación móvil, se puede calcular el porcentaje de dispositivos que se abordan mediante pruebas de compatibilidad, lo que podría identificar brechas de cobertura.

El diseño y la ejecución de pruebas no funcionales pueden involucrar habilidades o conocimientos especiales, como el conocimiento de las debilidades inherentes de un diseño o tecnología (p. ej., vulnerabilidades de seguridad asociadas con lenguajes de programación específicos) o la base de usuario específica (p. ej., las personas de los usuarios de sistemas de gestión de instalaciones de asistencia médica).

Consulte ISTQB Programa de Estudio – Nivel Avanzado – Analista de Pruebas, ISTQB-ATTA Programa de Estudio – Nivel Avanzado – Analista de Pruebas Técnicas, ISTQB-SEC Programa de Estudio – Nivel Avanzado – Pruebas de Seguridad y otros módulos de la ISTQB para especialistas para obtener más detalles sobre la prueba de las características de calidad no funcionales.

2.3.3 Pruebas de Caja Blanca

Las pruebas de caja blanca derivan pruebas basadas en la estructura interna o la implementación del sistema). La estructura interna puede incluir código, arquitectura, flujos de trabajo y/o flujos de datos dentro del sistema (consulte la sección 4.3).

La minuciosidad de las pruebas de caja blanca se puede medir a través de la cobertura estructural. La cobertura estructural es la medida en que se ha ejercido algún tipo de elemento estructural mediante pruebas, y se expresa como un porcentaje del tipo de elemento que se está cubriendo.

En el nivel de prueba de componentes, la cobertura de código se basa en el porcentaje de código de componente que se ha probado y se puede medir en términos de diferentes aspectos del código (elementos de cobertura), como el porcentaje de sentencias ejecutables probadas en el componente, o el porcentaje de resultados de decisión probados. Estos tipos de cobertura se denominan colectivamente cobertura de código. En el nivel de prueba de integración de componentes, las pruebas de caja blanca pueden basarse en la arquitectura del sistema, como son las interfaces entre componentes, y la cobertura estructural puede medirse en términos del porcentaje de interfaces ejercidas por las pruebas.

El diseño y la ejecución de pruebas de caja blanca pueden involucrar habilidades o conocimientos especiales, como la forma en que se construye el código (p. ej., para usar herramientas de cobertura de códigos), cómo se almacenan los datos (p. ej., para evaluar posibles consultas de bases de datos) y cómo usar herramientas de cobertura e interpretar correctamente sus resultados.

2.3.4 Pruebas relacionadas con el cambio

Cuando se realizan cambios en un sistema, ya sea para corregir un defecto o debido a una funcionalidad nueva o cambiante, se deben realizar pruebas para confirmar que los cambios han corregido el defecto o implementado la funcionalidad correctamente, y no han causado consecuencias adversas imprevistas.

- Pruebas de confirmación: Una vez que se soluciona un defecto, se puede probar el software con todos los casos de prueba que fallaron debido al defecto, que deben volver a ejecutarse en la nueva versión del software. El software también puede probarse con nuevas pruebas si, por ejemplo, faltaba la funcionalidad del defecto. Como mínimo, los pasos para reproducir los fallos causados por el defecto deben volver a ejecutarse en la nueva versión del software. El propósito de una prueba de confirmación es confirmar si el defecto original se ha solucionado correctamente.
- Pruebas de regresión: Es posible que un cambio realizado en una parte del código, ya sea una solución u otro tipo de cambio, pueda afectar accidentalmente el comportamiento de otras partes del código, ya sea dentro del mismo componente, en otros componentes del mismo sistema, o incluso en otros sistemas. Los cambios pueden incluir cambios en el entorno, como una nueva versión de un sistema operativo o un sistema de administración de bases de datos. Tales efectos secundarios no deseados se denominan regresiones. Las pruebas de regresión involucran la ejecución de pruebas para detectar dichos efectos secundarios no deseados.

Las pruebas de confirmación y las pruebas de regresión se realizan en todos los niveles de prueba.

Especialmente en los ciclos de vida de desarrollo iterativo e incremental (p. ej., Ágil), las nuevas prestaciones, los cambios en las prestaciones existentes y la refactorización del código dan como resultado cambios frecuentes en el código, que también requieren pruebas relacionadas con el cambio. Debido a la naturaleza evolutiva del sistema, las pruebas de confirmación y regresión son muy importantes. Esto es particularmente importante para los sistemas de Internet de las cosas donde los objetos individuales (p. ej., los dispositivos) se actualizan o reemplazan con frecuencia.

Los juegos de pruebas de regresión se ejecutan muchas veces y generalmente evolucionan lentamente, por lo que las pruebas de regresión son un fuerte candidato para la automatización. La automatización de estas pruebas debe comenzar al principio del proyecto (ver capítulo 6).

2.3.5 Tipos y niveles de prueba

Es posible realizar cualquiera de los tipos de prueba mencionados anteriormente en cualquier nivel de prueba. Para ilustrarlo, se darán ejemplos de pruebas funcionales, no funcionales, de caja blanca y relacionados con cambios en todos los niveles de prueba para una aplicación bancaria, comenzando con pruebas funcionales:

- Para las pruebas de componente, las pruebas se diseñan en función de cómo un componente debe calcular el interés compuesto.
- Para las pruebas de integración de componentes, las pruebas se diseñan en función de cómo la información de la cuenta capturada en la interfaz de usuario se pasa a la lógica comercial.
- Para las pruebas de sistema, las pruebas se diseñan en función de cómo los titulares de cuentas pueden solicitar una línea de crédito en sus cuentas corrientes.
- Para las pruebas de integración del sistema, las pruebas se diseñan en función de cómo el sistema utiliza un microservicio externo para verificar la calificación crediticia del titular de una cuenta.
- Para las pruebas de aceptación, las pruebas se diseñan según la forma en que el banquero se encarga de aprobar o rechazar una solicitud de crédito.

Los siguientes son ejemplos de pruebas no funcionales:

- Para las pruebas de componente, las pruebas de rendimiento están diseñadas para evaluar la cantidad de ciclos de CPU necesarios para realizar un cálculo de interés total complejo.

- Para las pruebas de integración de componentes, las pruebas de seguridad están diseñadas para vulnerabilidades de desbordamiento de búfer debido a los datos que pasan desde la interfaz de usuario a la lógica comercial.
- Para las pruebas de sistema, las pruebas de portabilidad están diseñadas para verificar si la capa de presentación funciona en todos los navegadores y dispositivos móviles compatibles.
- Para las pruebas de integración del sistema, las pruebas de confiabilidad están diseñadas para evaluar la solidez del sistema, si el microservicio de calificación crediticia no responde.
- Para las pruebas de aceptación, las pruebas de usabilidad están diseñadas para evaluar la accesibilidad de la interfaz de procesamiento de crédito del banquero para personas con discapacidades.

Los siguientes son ejemplos de pruebas de caja blanca:

- Para las pruebas de componente, las pruebas están diseñadas para lograr una cobertura completa de sentencias y de decisiones (consulte la sección 4.3) para todos los componentes que realizan cálculos financieros.
- Para las pruebas de integración de componentes, las pruebas están diseñadas para ejercer la forma en que cada pantalla en la interfaz del navegador pasa los datos a la siguiente pantalla y a la lógica comercial.
- Para las pruebas de sistema, las pruebas están diseñadas para cubrir secuencias de páginas Web que pueden ocurrir durante una aplicación de línea de crédito.
- Para las pruebas de integración del sistema, las pruebas están diseñadas para ejercer todos los tipos de consultas posibles enviadas al microservicio de calificación crediticia.
- Para las pruebas de aceptación, las pruebas están diseñadas para cubrir todas las estructuras de archivos de datos financieros compatibles y los rangos de valores para las transferencias de banco a banco.

Por último, los siguientes son ejemplos de pruebas relacionadas con el cambio:

- Para las pruebas de componente, las pruebas de regresión automatizadas se crean para cada componente y se incluyen dentro del marco de integración continua.
- Para las pruebas de integración de componente, las pruebas están diseñadas para confirmar correcciones a defectos relacionados con la interfaz a medida que las correcciones se verifican en el repositorio de códigos.
- Para las pruebas de sistema, todas las pruebas para un flujo de trabajo dado se vuelven a ejecutar si cambia alguna pantalla en ese flujo de trabajo.
- Para las pruebas de integración del sistema, las pruebas de la aplicación que interactúan con el microservicio de calificación crediticia se vuelven a ejecutar diariamente como parte del despliegue continuo de ese microservicio.
- Para las pruebas de aceptación, todas las pruebas fallidas previamente se vuelven a ejecutar después de que se solucione un defecto detectado en las pruebas de aceptación.

Si bien esta sección proporciona ejemplos de cada tipo de prueba en todos los niveles, no es necesario, para todo el software, tener cada tipo de prueba representado en cada nivel. Sin embargo, es importante ejecutar los tipos de prueba aplicables en cada nivel, especialmente el nivel más temprano donde se produce el tipo de prueba.

2.4 Pruebas de mantenimiento

Una vez implementado en los entornos de producción, el software y los sistemas se deben mantener. Los cambios de varios tipos son casi inevitables en el software y los sistemas entregados, ya sea para corregir los defectos detectados en el uso operacional, para agregar una nueva funcionalidad, o para eliminar o alterar una funcionalidad ya entregada. El mantenimiento también es necesario para preservar o mejorar las características de calidad no funcionales del componente o sistema durante su vida útil, especialmente la eficiencia de rendimiento, compatibilidad, confiabilidad, seguridad, y portabilidad.

Cuando se realizan cambios como parte del mantenimiento, se deben realizar pruebas de mantenimiento, tanto para evaluar el éxito con el que se realizaron los cambios como para detectar posibles efectos secundarios (p. ej., regresiones) en partes del sistema que permanecen sin cambios (que suele ser la mayor parte del sistema).
Las pruebas de mantenimiento se centran en probar los cambios en el sistema, así como en probar partes no modificadas que podrían haber sido afectadas por los cambios. El mantenimiento puede implicar entregas planificadas y entregas no planificadas (correcciones en caliente).

Una entrega de mantenimiento puede requerir pruebas de mantenimiento en múltiples niveles de prueba, utilizando varios tipos de prueba, según su alcance. El alcance de las pruebas de mantenimiento depende de:

- El grado de riesgo del cambio, por ejemplo, el grado en que el área cambiada del software se comunica con otros componentes o sistemas
- El tamaño del sistema existente
- El tamaño del cambio

2.4.1 Causas para el mantenimiento

Hay varias razones por las que se realiza el mantenimiento del software y, por lo tanto, pruebas de mantenimiento tanto para cambios planificados como no planificados.

Podemos clasificar las causas para el mantenimiento de la siguiente manera:

- Modificación, tales como mejoras planificadas (p. ej., basadas en versiones), cambios correctivos y de emergencia, cambios en el entorno operativo (como las actualizaciones planificadas de sistemas operativos o bases de datos), actualizaciones del software COTS y parches para defectos y vulnerabilidades.
- La migración, como la de una plataforma a otra, que puede requerir pruebas operativas del nuevo entorno así como del software modificado, o pruebas de conversión de datos cuando los datos de otra aplicación se migrarán al sistema que se está manteniendo.
- Retiro, como cuando una aplicación llega al final de su vida útil.

Cuando se retira una aplicación o un sistema, esta puede requerir pruebas de migración o de archivo de datos si se requieren largos períodos de retención de datos. También puede ser necesario probar los procedimientos de restauración/recuperación después de archivar durante largos períodos de retención. Además, es posible que se necesiten pruebas de regresión para garantizar que cualquier funcionalidad que permanezca en servicio aún funcione.

Para los sistemas de Internet de las cosas, las pruebas de mantenimiento pueden iniciarse mediante la introducción de elementos completamente nuevos o modificados, como dispositivos de hardware y servicios de software, en el sistema general. Las pruebas de mantenimiento para dichos sistemas hacen especial hincapié en las pruebas de integración a diferentes niveles (p. ej., nivel de red, nivel de aplicación) y en aspectos de seguridad, en particular los relacionados con datos personales.

2.4.2 Análisis del impacto para el mantenimiento

El análisis de impacto evalúa los cambios que se realizaron para una entrega de mantenimiento para identificar las consecuencias previstas, así como los efectos secundarios esperados y posibles de un cambio, y para identificar las áreas en el sistema que se verán afectadas por el cambio. El análisis de impacto también puede ayudar a identificar el impacto de un cambio en las pruebas existentes. Los efectos secundarios y las áreas afectadas en el sistema deben someterse a pruebas de regresión, posiblemente después de actualizar cualquier prueba existente afectada por el cambio.

El análisis del impacto puede realizarse antes de que se realice un cambio, para ayudar a decidir si el cambio se debe realizar, en función de las posibles consecuencias en otras áreas del sistema.

El análisis del impacto puede ser difícil si:

- Las especificaciones (p. ej., requisitos comerciales, historias de usuario, arquitectura) están desactualizadas o faltan
- Los casos de prueba no están documentados o están desactualizados
- La trazabilidad bidireccional entre pruebas y la base de pruebas no se ha mantenido
- El soporte para las herramientas es débil o inexistente
- Las personas involucradas no tienen conocimiento de dominio y/o del sistema
- Se ha prestado poca atención a la capacidad de mantenimiento del software durante el desarrollo

3 Pruebas Estáticas — 135 minutos

Palabras clave

revisión ad hoc, revisión basada en listas de verificación, pruebas dinámicas, revisión formal, revisión informal, inspección, lectura basada en la perspectiva, revisión, revisión basada en roles, revisión basada en escenarios, análisis estático, pruebas estáticas, revisión técnica, revisión guiada

Objetivos de Aprendizaje para las Pruebas Estáticas

3.1 Conceptos Básicos de las Pruebas Estáticas

FL-3.1.1 (K1) Reconocer los tipos de productos de software que se pueden examinar mediante las diferentes técnicas de prueba estática

FL-3.1.2 (K2) Usar ejemplos para describir el valor de las pruebas estáticas

FL-3.1.3 (K2) Explicar la diferencia entre las técnicas estáticas y dinámicas, teniendo en cuenta los objetivos, los tipos de defectos que deben identificarse y la función de estas técnicas en el ciclo de vida del software

3.2 Proceso de revisión

FL-3.2.1 (K2) Resumir las actividades del proceso de revisión del producto de trabajo

FL-3.2.2 (K1) Reconocer los diferentes roles y responsabilidades en una revisión formal

FL-3.2.3 (K2) Explicar las diferencias entre los diferentes tipos de revisiones: revisión informal, revisión guiada, revisión técnica, e inspección

FL-3.2.4 (K3) Aplicar una técnica de revisión a un producto de trabajo para encontrar defectos

FL-3.2.5 (K2) Explicar los factores que contribuyen a una revisión exitosa

3.1 Conceptos Básicos de las Pruebas Estáticas

En contraste con las pruebas dinámicas, que requieren la ejecución del software que se está probando, las pruebas estáticas se basan en el examen manual de los productos de trabajo (es decir, revisiones) o la evaluación del código u otros productos de trabajo (es decir, el análisis estático). Ambos tipos de pruebas estáticas evalúan el código u otro producto de trabajo que se está probando sin ejecutar realmente el código o el producto de trabajo que se está probando.

El análisis estático es importante para los sistemas informáticos críticos para la seguridad (p. ej., software de aviación, médico o nuclear), pero el análisis estático también se ha vuelto importante y común en otros entornos. Por ejemplo, el análisis estático es una parte importante de las pruebas de seguridad. El análisis estático también suele incorporarse a los sistemas automatizados de construcción y entrega, por ejemplo, en el desarrollo Ágil, la entrega continua y la implementación continua.

3.1.1 Productos de trabajo que se pueden examinar mediante las pruebas estáticas

Casi cualquier producto de trabajo se puede examinar utilizando pruebas estáticas (revisiones y/o análisis estático), por ejemplo:

- Especificaciones, incluidos los requisitos comerciales, los requisitos funcionales y los requisitos de seguridad
- Epopeyas, historias de usuario, y criterios de aceptación
- Especificaciones de arquitectura y diseño
- Código
- Productos de prueba, incluidos planes de prueba, casos de prueba, procedimientos de prueba y giones de prueba automatizados
- Guías del usuario
- Páginas Web
- Contratos, planes de proyectos, cronogramas y presupuestos
- Modelos, tales como diagramas de actividad, que se pueden usar para pruebas basadas en modelos (ver ISTQB Programa de Estudio – Nivel Básico – Extensión – Pruebas Basadas en Modelos y Kramer 2016)

Las revisiones pueden aplicarse a cualquier producto de trabajo que los participantes sepan leer y comprender. El análisis estático se puede aplicar de manera eficiente a cualquier producto de trabajo con una estructura formal (generalmente código o modelos) para los cuales existe una herramienta de análisis estático adecuada. El análisis estático incluso se puede aplicar con herramientas que evalúan los productos de trabajo escritos en lenguaje natural, como los requisitos (p. ej., revisión de la ortografía, gramática y legibilidad).

3.1.2 Beneficios de las pruebas estáticas

Las técnicas de prueba estática proporcionan una variedad de beneficios. Cuando se aplica temprano en el ciclo de vida del desarrollo del software, las pruebas estáticas permiten la detección temprana de defectos antes de que se realicen las pruebas dinámicas (p. ej., en revisiones de requisitos o especificaciones de diseño, refinamiento de la cartera de pedidos de productos, etc.). Los defectos que se detectan temprano a menudo son mucho más económicos de eliminar que los defectos que se encuentran más adelante en el ciclo de vida, especialmente en comparación con los defectos encontrados después de la implementación del software y su uso activo. El uso de técnicas de prueba estática para

encontrar defectos y luego corregirlos rápidamente es casi siempre más barato para la organización que usar pruebas dinámicas para encontrar defectos en el objeto de prueba y luego corregirlos, especialmente al considerar los costos adicionales asociados con la actualización de otros productos de trabajo y realizar pruebas de confirmación y regresión.

Los beneficios adicionales de las pruebas estáticas pueden incluir:

- Detectar y corregir defectos de manera más eficiente y antes de la ejecución dinámica de la prueba
- Identificar defectos que no se encuentran fácilmente mediante pruebas dinámicas
- Prevenir defectos en el diseño o la codificación al descubrir inconsistencias, ambigüedades, contradicciones, omisiones, imprecisiones y redundancias en los requisitos
- Incremento de la productividad de desarrollo (p. ej., debido a un diseño mejorado, código más mantenible)
- Reducción de costos y tiempos de desarrollo
- Reducción de costos y tiempo de prueba
- Reducción del costo total de la calidad a lo largo de la vida útil del software, debido a menos fallos más adelante en el ciclo de vida o después de la entrega en operación
- Mejorar la comunicación entre los miembros del equipo mientras participan en las revisiones

3.1.3 Diferencias entre las pruebas estáticas y las pruebas dinámicas

Las pruebas estáticas y las pruebas dinámicas pueden tener los mismos objetivos (consulte la sección 1.1.1), como son proporcionar una evaluación de la calidad de los productos de trabajo e identificar defectos lo antes posible. Las pruebas estáticas y dinámicas se complementan al encontrar diferentes tipos de defectos.

Una distinción principal es que las pruebas estáticas encuentran defectos en los productos de trabajo directamente en lugar de identificar fallos causados por defectos cuando se ejecuta el software. Un defecto puede residir en un producto de trabajo durante mucho tiempo sin causar un fallo. El camino donde se encuentra el defecto puede ser raramente ejercitado o difícil de alcanzar, por lo que no será fácil construir y ejecutar una prueba dinámica que lo encuentre. Las pruebas estáticas pueden encontrar el defecto con mucho menos esfuerzo.

Otra distinción es que las pruebas estáticas se pueden usar para mejorar la consistencia y la calidad interna de los productos de trabajo, mientras que las pruebas dinámicas generalmente se enfocan en comportamientos visibles externamente.

En comparación con las pruebas dinámicas, los defectos típicos que son más fáciles y económicos de encontrar y corregir a través de las pruebas estáticas incluyen:

- Defectos de requisitos (p. ej., inconsistencias, ambigüedades, contradicciones, omisiones, inexactitudes y redundancias)
- Defectos de diseño (p. ej., algoritmos ineficientes o estructuras de base de datos, alto acoplamiento, baja cohesión)
- Defectos de codificación (p. ej., variables con valores indefinidos, variables que se declaran pero nunca se usan, código inalcanzable, código duplicado)
- Desviaciones de las normas (p. ej., falta de cumplimiento con las normas de codificación)
- Especificaciones de interfaz incorrectas (p. ej., diferentes unidades de medida utilizadas más por el sistema que llama que por el sistema llamado)
- Vulnerabilidades de seguridad (p. ej., susceptibilidad a desbordamientos de búfer)
- Brechas o inexactitudes en la trazabilidad o cobertura de la base de la prueba (p. ej., pruebas

faltantes para un criterio de aceptación)

Además, la mayoría de los tipos de defectos de mantenimiento solo se pueden encontrar mediante pruebas estáticas (p. ej., modularización inadecuada, reutilización deficiente de los componentes, código que es difícil de analizar y modificar sin introducir nuevos defectos).

3.2 Proceso de revisión

Las revisiones varían de informal a formal. Las revisiones informales se caracterizan por no seguir un proceso definido y no tener resultados documentados formales. Las revisiones formales se caracterizan por la participación del equipo, los resultados documentados de la revisión y los procedimientos documentados para llevar a cabo la revisión. La formalidad de un proceso de revisión está relacionada con factores tales como el modelo de ciclo de vida del desarrollo del software, la madurez del proceso de desarrollo, la complejidad del producto de trabajo que se debe revisar, cualquier requisito legal o normativo, y/o la necesidad de una ruta de revisión.

El enfoque de una revisión depende de los objetivos de la revisión acordados (p. ej., encontrar defectos, adquirir comprensión, educar a los participantes, como probadores y nuevos miembros del equipo, o discutir y decidir por consenso).

La norma ISO (ISO/IEC 20246) contiene descripciones más detalladas del proceso de revisión de los productos de trabajo, incluidas las funciones y las técnicas de revisión.

3.2.1 Proceso de revisión del producto de trabajo

El proceso de revisión comprende las siguientes actividades principales:

Planificación

- Definir el alcance, que incluye el propósito de la revisión, qué documentos o partes de documentos se deben revisar y las características de calidad que se evaluarán
- Estimar el esfuerzo y período de tiempo
- Identificar las características de la revisión, como son el tipo de revisión con roles, actividades y listas de comprobación
- Seleccionar las personas para participar en la revisión y asignación de roles
- Definir los criterios de entrada y salida para tipos de revisión más formales (p. ej., inspecciones)
- Verificar que se cumplan los criterios de entrada (para tipos de revisión más formales)

Iniciar la revisión

- Distribuir el producto de trabajo (físicamente o por medios electrónicos) y otro material, como son emitir formularios de registro, listas de verificación y productos de trabajo relacionados
- Explicar a los participantes el alcance, los objetivos, el proceso, los roles y los productos de trabajo
- Responder a cualquier pregunta que los participantes puedan tener sobre la revisión

Revisión individual (es decir, preparación individual)

- Revisar todo o parte del producto de trabajo
- Observar los defectos potenciales, recomendaciones y preguntas

Comunicación y análisis de problemas

- Comunicar defectos potenciales identificados (p. ej., en una reunión de revisión)
- Analizar posibles defectos, asignándoles la propiedad y el estado
- Evaluar y documentar las características de calidad

- Evaluar los resultados de la revisión en comparación con los criterios de salida para tomar una decisión de revisión (rechazar; cambios importantes necesarios; aceptar, posiblemente con cambios menores)

Corrección e información de la prueba

- Crear informes de defectos para aquellos hallazgos que requieran cambios
- Corregir los defectos encontrados (generalmente realizados por el autor) en el producto de trabajo revisado
- Comunicar los defectos a la persona o equipo apropiado (cuando se encuentra en un producto de trabajo relacionado con el producto de trabajo revisado)
- Registrar el estado actualizado de los defectos (en revisiones formales), que posiblemente incluya el acuerdo del autor del comentario
- Recopilar las métricas (para tipos de revisión más formales)
- Verificar que se cumplan los criterios de salida (para tipos de revisión más formales)
- Aceptar el producto de trabajo cuando se alcanzan los criterios de salida

Los resultados de una revisión del producto de trabajo varían, según el tipo de revisión y la formalidad, como se describe en la sección 3.2.3.

3.2.2 Funciones y responsabilidades en una revisión formal

Una revisión formal típica incluirá los siguientes roles:

Autor

- Crea el producto de trabajo que se revisa
- Corrige defectos en el producto de trabajo bajo revisión (si es necesario)

Administración

- Es responsable de la planificación de la revisión
- Decide sobre la ejecución de revisiones
- Asigna personal, presupuesto y tiempo
- Supervisa la rentabilidad en curso
- Ejecuta decisiones de control en caso de resultados inadecuados

Facilitador (frecuentemente llamado moderador)

- Asegura la ejecución efectiva de las reuniones de revisión (cuando se efectúan)
- Media, si es necesario, entre los distintos puntos de vista
- A menudo, es la persona sobre la cuál descansa el éxito de la revisión

Líder de la revisión

- Asume la responsabilidad total de la revisión
- Decide quién participará y organiza cuándo y dónde tendrá lugar

Revisores

- Pueden ser expertos en la materia, personas que trabajan en el proyecto, partes interesadas con interés en el producto de trabajo y/o personas con historial técnico o empresarial específico
- Identifican defectos potenciales en el producto de trabajo que se revisa
- Pueden representar diferentes perspectivas (p. ej., programador, usuario, operador, analista de empresas, experto en usabilidad, etc.)

Escriba (o registrador)

- Coteja los posibles defectos encontrados durante la actividad de revisión individual
- Records new potential defects, open points, and decisions from the review meeting (when held)

Registra nuevos defectos potenciales, puntos abiertos y decisiones de la reunión de revisión (cuando se efectúa. En algunos tipos de revisión, una persona puede desempeñar más de una función, y las acciones asociadas con cada función también pueden variar según el tipo de revisión. Además, con la llegada de las herramientas para respaldar el proceso de revisión, especialmente el registro de defectos, puntos abiertos y decisiones, a veces no hay necesidad de un escriba.

Por otra parte, es posible realizar funciones más detalladas, como se describe en la norma ISO (ISO/IEC 20246).

3.2.3 Tipos de revisión

Aunque las revisiones se pueden utilizar para varios propósitos, uno de los objetivos principales es descubrir defectos. Todos los tipos de revisión pueden ayudar en la detección de defectos, y el tipo de revisión seleccionado debe basarse en las necesidades del proyecto, los recursos disponibles, el tipo de producto y los riesgos, el dominio empresarial y la cultura de la empresa, entre otros criterios de selección.

Las revisiones se pueden clasificar de acuerdo a varios atributos. A continuación se enumeran los cuatro tipos de revisiones más comunes y sus atributos asociados.

Revisión informal (p. ej., verificación de amigos, emparejamiento, revisión de pares)

- Objetivo principal: Detectar posibles defectos
- Posibles propósitos adicionales: Generar nuevas ideas o soluciones, resolver rápidamente problemas menores
- No se basa en un proceso formal (documentado)
- Puede no implicar una reunión de revisión
- Lo puede realizar un colega del autor (control del compañero) o más personas
- Los resultados pueden ser documentados
- Varía en su utilidad dependiendo de los revisores
- Uso opcional de listas de comprobación
- Muy usado en desarrollo Ágil

Revisión guiada

- Principales propósitos: Encontrar defectos, mejorar el producto de software, considerar implementaciones alternativas, evaluar el cumplimiento de normas y especificaciones
- Posibles propósitos adicionales: Intercambiar ideas sobre técnicas o variaciones de estilo, capacitación de participantes, lograr consenso
- La preparación individual antes de la reunión de revisión es opcional
- La reunión de revisión suele ser dirigida por el autor del producto de trabajo
- La participación de un escriba es obligatoria
- Uso opcional de listas de comprobación
- Puede tomar la forma de escenarios, ejecuciones en seco o simulaciones
- Se pueden producir posibles registros de defectos e informes de revisión
- Puede variar en la práctica de bastante informal a muy formal

Revisión técnica

- Objetivo principal: Obtener consenso, detectar posibles defectos
- Posibles propósitos adicionales: Evaluar la calidad y generar confianza en el producto de trabajo, generar nuevas ideas, motivar y permitir a los autores mejorar los futuros productos de trabajo, considerar implementaciones alternativas
- Los revisores deben ser pares técnicos del autor y expertos técnicos en la misma o en otras disciplinas
- Se requiere preparación individual antes de la reunión de revisión
- La reunión de revisión es opcional, idealmente dirigida por un facilitador capacitado (normalmente no es el autor)
- La participación de un escriba es obligatoria, idealmente no el autor
- Uso opcional de listas de comprobación
- Registros de defectos potenciales e informes de la revisión se producen normalmente

Inspección

- Principales objetivos: Detectar defectos potenciales, evaluar la calidad y generar confianza en el producto de trabajo, prevenir defectos similares en el futuro a través del aprendizaje de los autores y el análisis de la causa raíz
- Posibles propósitos adicionales: Motivar y permitir a los autores mejorar los futuros productos de trabajo y el proceso de desarrollo de software, logrando el consenso
- Sigue un proceso definido con resultados documentados formales, basados en reglas y listas de comprobación
- Utiliza roles claramente definidos, como los especificados en la sección 3.2.2 que son obligatorios, y puede incluir un lector dedicado (que lee el producto del trabajo en voz alta durante la reunión de revisión)
- Es necesaria la preparación individual antes de la reunión de revisión
- Los revisores son colegas del autor o expertos en otras disciplinas que son relevantes para el producto de trabajo

- Criterios de entrada y salida especificados
- La participación de un escriba es obligatoria
- La reunión de revisión es dirigida por un facilitador capacitado (normalmente no es el autor)
- El autor no puede actuar como líder de la revisión, lector o escriba
- Se producen posibles registros de defectos e informes de revisión
- Las métricas se recopilan y se utilizan para mejorar todo el proceso de desarrollo de software, incluido el proceso de inspección

Un solo producto de trabajo puede ser objeto de más de un tipo de revisión. Si se utiliza más de un tipo de revisión, el orden puede variar. Por ejemplo, se puede llevar a cabo una revisión informal antes de una revisión técnica, para garantizar que el producto de trabajo esté listo para una revisión técnica.

Los tipos de revisiones descritos anteriormente se pueden hacer como revisiones por pares, es decir, realizadas por colegas en un nivel organizativo aproximado similar.

Los tipos de defectos encontrados en una revisión varían, dependiendo especialmente del producto de trabajo que se está revisando. Consulte la sección 3.1.3 para ver ejemplos de defectos que pueden encontrarse en las revisiones de diferentes productos de trabajo, y consulte Gilb 1993 para obtener información sobre inspecciones formales.

3.2.4 Aplicación de técnicas de revisión

Existen varias técnicas de revisión que pueden aplicarse durante la actividad de revisión individual (es decir, preparación individual) para descubrir defectos. Estas técnicas se pueden utilizar en los tipos de revisión descritos anteriormente. La efectividad de las técnicas puede variar según el tipo de revisión utilizada. A continuación, se enumeran ejemplos de diferentes técnicas de revisión individual para varios tipos de revisión.

Ad hoc

En una revisión ad hoc, los revisores reciben poca o ninguna orientación sobre cómo se debe realizar esta tarea. Los revisores a menudo leen el producto de trabajo de forma secuencial, identificando y documentando los problemas a medida que los encuentran. La revisión ad hoc es una técnica comúnmente utilizada que requiere poca preparación. Esta técnica es muy dependiente de las habilidades del revisor y puede conllevar a que muchos problemas sean reportados por duplicado por revisores diferentes.

Basada en listas de comprobación

Una revisión basada en listas de comprobación es una técnica sistemática, en la cual los revisores detectan problemas basados en listas de comprobación que se distribuyen al inicio de la revisión (p. ej., por el facilitador). Una revisión basada en lista de comprobación consiste en un conjunto de preguntas basadas en defectos potenciales, que pueden derivarse de la experiencia. Las listas de comprobación deben ser específicas para el tipo de producto de trabajo que se revisa y deben mantenerse regularmente para cubrir los tipos de problemas que faltaron en revisiones anteriores. La principal ventaja de la técnica basada en listas de verificación es una cobertura sistemática de los tipos típicos de defectos. Se debe tener cuidado de no seguir simplemente la lista de comprobación en la revisión individual, sino también de buscar defectos fuera de la lista de comprobación.

Escenarios y ejecuciones en seco

En una revisión basada en escenarios, los revisores reciben pautas estructuradas sobre cómo leer todo el producto de trabajo. Un enfoque basado en escenarios ayuda a los revisores a realizar “ejecuciones en seco” en el producto de trabajo en función del uso esperado del producto de trabajo (si el producto de trabajo está documentado en un formato adecuado, como los casos de uso). Estos escenarios proporcionan a los revisores mejores pautas sobre cómo identificar tipos de defectos específicos que las simples entradas de la

lista de comprobación.

Al igual que con las revisiones basadas en listas de comprobación, para no perder otros tipos de defectos (p. ej., las prestaciones que faltan), los revisores no deben estar limitados a los escenarios documentados.

Basada en roles

Una revisión basada en roles es una técnica en la que los revisores evalúan el producto de trabajo desde la perspectiva de los roles de las partes interesadas individuales. Los roles típicos incluyen tipos de usuarios finales específicos (con experiencia, sin experiencia, mayores, hijo, etc.) y roles específicos en la organización (administrador de usuarios, administrador del sistema, probador de rendimiento, etc.).

Basada en la perspectiva

En la lectura basada en la perspectiva, similar a una revisión basada en roles, los revisores adoptan diferentes puntos de vista de las partes interesadas en la revisión individual. Los puntos de vista típicos de las partes interesadas incluyen usuario final, comercialización, diseñador, probador u operaciones. El uso de diferentes puntos de vista de las partes interesadas lleva a una mayor profundidad en la revisión individual con menos duplicación de problemas entre los revisores.

Además, la lectura basada en perspectivas también requiere que los revisores intenten utilizar el producto de trabajo bajo revisión para generar el producto que obtendrían del mismo. Por ejemplo, un probador intentaría generar un borrador de pruebas de aceptación si realiza una lectura basada en perspectivas en una especificación de requisitos para ver si se incluyó toda la información necesaria. Además, en la lectura basada en perspectivas, se espera que se usen listas de comprobación.

Los estudios empíricos han demostrado que la lectura basada en perspectivas es la técnica general más efectiva para revisar los requisitos y los productos de trabajo técnico. Un factor clave de éxito es incluir y sopesar los puntos de vista de diferentes partes interesadas de manera adecuada, en función de los riesgos. Consulte Shul 2000 para obtener detalles sobre la lectura basada en perspectivas y Sauer 2000 para conocer la efectividad de los diferentes tipos de revisión.

3.2.5 Factores de éxito para las revisiones

Para tener una revisión exitosa, se debe considerar el tipo apropiado de revisión y las técnicas utilizadas. Además, hay una serie de otros factores que afectarán el resultado de la revisión.

Los factores de éxito organizacional para las revisiones incluyen:

- Cada revisión tiene objetivos claros, definidos durante la planificación de la revisión y utilizados como criterios de salida medibles.
- Se aplican tipos de revisión que son adecuadas para lograr los objetivos y para el tipo y nivel de productos de trabajo de software y participantes.
- Cualquier técnica de revisión utilizada, como la revisión basada en listas de comprobación o basada en roles, es adecuada para la identificación efectiva de defectos en el producto de trabajo que se revisará.
- Las listas de comprobación utilizadas abordan los principales riesgos y están actualizadas.
- Los documentos grandes se escriben y revisan en pequeños fragmentos, de modo que el control de calidad se ejerce al proporcionar a los autores comentarios tempranos y frecuentes sobre los defectos.
- Los participantes tienen tiempo suficiente para prepararse.
- Las revisiones están programadas con la debida antelación.
- La administración respalda el proceso de revisión (p. ej., al incorporar el tiempo adecuado para las actividades de revisión en los cronogramas del proyecto).

Los factores de éxito para las revisiones relacionados con las personas incluyen:

- Las personas correctas están involucradas para cumplir con los objetivos de la revisión, por ejemplo, personas con diferentes habilidades o perspectivas, que pueden usar el documento como información de trabajo.
- Los probadores son considerados valiosos revisores que contribuyen a la revisión y aprenden sobre el producto de trabajo, lo que les permite preparar pruebas más efectivas y preparar esas pruebas con antelación.
- Los participantes dedican tiempo y atención adecuada al detalle.
- Las revisiones se llevan a cabo en partes pequeñas, de manera que los revisores no pierdan la concentración durante la revisión individual y/o en la reunión de revisión (cuando se efectúe).
- Los defectos encontrados son reconocidos, apreciados y manejados de manera objetiva.
- La reunión está bien gestionada, por lo que los participantes consideran que es un uso valioso de su tiempo.
- La revisión se realiza en una atmósfera de confianza; el resultado no se utilizará para la evaluación de los participantes.
- Los participantes evitan el lenguaje corporal y los comportamientos que puedan indicar aburrimiento, exasperación u hostilidad hacia otros participantes.
- Se proporciona capacitación adecuada, especialmente para tipos de revisión más formales, como son las inspecciones.
- Se promueve una cultura de aprendizaje y mejora de procesos.

Consulte a Gilb 1993, Wiegers 2002 y Van Veenendaal 2004 para obtener más información sobre las revisiones exitosas.

4 Técnicas de Pruebas — 330 minutos

Palabras clave

técnica de prueba de caja negra, análisis de valor límite, pruebas basadas en listas de comprobación, cobertura, cobertura de decisiones, prueba de tabla de decisiones, predicción de errores, segmentación de equivalencia, técnica de prueba basada en la experiencia, prueba exploratoria, prueba de transición de estado, cobertura de sentencia, técnica de prueba, prueba de casos de uso, técnica de caja blanca

Objetivos de Aprendizaje para las Técnicas de Prueba

4.1 Categorías de técnicas de prueba

FL-4.1.1 (K2) Explicar las características, las similitudes y las diferencias entre las técnicas de caja negra, las técnicas de caja blanca y las técnicas de pruebas basadas en la experiencia

4.2 Técnicas de prueba de caja negra

FL-4.2.1 (K3) Aplicar la segmentación de equivalencia para derivar los casos de prueba a partir de requisitos dados

FL-4.2.2 (K3) Aplicar el análisis del valor límite para derivar los casos de prueba a partir de requisitos dados

FL-4.2.3 (K3) Aplicar pruebas de tabla de decisiones para derivar casos de prueba a partir de requisitos dados

FL-4.2.4 (K3) Aplicar pruebas de transición de estado para derivar casos de prueba a partir de requisitos dados

FL-4.2.5 (K2) Explicar cómo derivar casos de prueba a partir de un caso de uso

4.3 Técnicas de prueba de caja blanca

FL-4.3.1 (K2) Explicar la cobertura de la sentencias

FL-4.3.2 (K2) Explicar la cobertura de decisiones

FL-4.3.3 (K2) Explicar el valor de las pruebas de sentencia y cobertura de decisiones

4.4 Técnicas de prueba basadas en la experiencia

FL-4.4.1 (K2) Explicar la predicción de errores

FL-4.4.2 (K2) Explicar las pruebas exploratorias

FL-4.4.3 (K2) Explicar las pruebas basadas en listas de comprobación

4.1 Categorías de técnicas de prueba

El propósito de una técnica de prueba, incluidas las analizadas en esta sección, es ayudar a identificar las condiciones de prueba, los casos de prueba y los datos de prueba.

4.1.1 Selección de las técnicas de prueba

La elección de qué técnicas de prueba usar depende de varios factores, entre los que se incluyen los siguientes:

- Tipo de componente o sistema
- Complejidad del componente o sistema
- Normas regulatorias
- Requisitos del cliente o contractuales
- Niveles de riesgo
- Tipos de riesgo
- Objetivos de la prueba
- Documentación disponible
- Conocimientos y habilidades del probador
- Herramientas disponibles
- Tiempo y presupuesto
- Modelo de ciclo de vida de desarrollo de software
- Uso previsto del software
- Experiencia previa con el uso de las técnicas de prueba en el componente o sistema que se va a probar
- Tipos de defectos esperados en el componente o sistema

Algunas técnicas son más aplicables a ciertas situaciones y niveles de prueba; otras son aplicables a todos los niveles de prueba. Al crear casos de prueba, los probadores generalmente usan una combinación de técnicas de prueba para lograr los mejores resultados del esfuerzo de prueba.

El uso de técnicas de prueba en el análisis de prueba, el diseño de prueba y las actividades de implementación de prueba pueden variar desde muy informales (poca o ninguna documentación) hasta muy formales. El nivel adecuado de formalidad depende del contexto de las pruebas, incluida la madurez de los procesos de prueba y desarrollo, las limitaciones de tiempo, la seguridad o los requisitos reglamentarios, el conocimiento y las habilidades de las personas involucradas y el modelo de ciclo de vida del desarrollo de software que se está siguiendo.

4.1.2 Categorías de las técnicas de prueba y sus características

En este programa de estudio, las técnicas de prueba se clasifican en caja negra, caja blanca o basadas en la experiencia.

Las técnicas de prueba de caja negra (también llamadas técnicas de comportamiento o basadas en el comportamiento) se basan en un análisis de la base de prueba adecuada (p. ej., documentos de requisitos formales, especificaciones, casos de uso, historias de usuario o procesos comerciales). Estas técnicas son aplicables tanto a las pruebas funcionales como a las no funcionales.

Las técnicas de prueba de caja negra se concentran en las entradas y salidas del objeto de prueba sin hacer referencia a su estructura interna. Las técnicas de prueba de caja blanca (también llamadas técnicas estructurales o basadas en la estructura) se basan en un análisis de la arquitectura, diseño detallado, estructura interna o el código del objeto de prueba. A diferencia de las técnicas de prueba de caja negra, las técnicas de prueba de caja blanca se concentran en la estructura y el procesamiento dentro del objeto de prueba.

Las técnicas de prueba basadas en la experiencia aprovechan la experiencia de los desarrolladores, probadores y usuarios para diseñar, implementar y ejecutar pruebas. Estas técnicas a menudo se combinan con técnicas de prueba de caja negra y caja blanca.

Las características comunes de las técnicas de prueba de caja negra incluyen las siguientes:

- Las condiciones de prueba, los casos de prueba y los datos de prueba se derivan de una base de prueba que puede incluir requisitos de software, especificaciones, casos de uso e historias de usuario
- Los casos de prueba se pueden usar para detectar brechas entre los requisitos y la implementación de los requisitos, así como las desviaciones de los mismos
- La cobertura se mide en función de los elementos probados en la base de prueba y la técnica aplicada a la base de prueba

Las características comunes de las técnicas de prueba de caja blanca incluyen las siguientes:

- Las condiciones de prueba, los casos de prueba y los datos de prueba se derivan de una base de prueba que puede incluir código, arquitectura de software, diseño detallado o cualquier otra fuente de información relacionada con la estructura del software
- La cobertura se mide en función de los elementos probados dentro de una estructura seleccionada (p. ej., el código o las interfaces)
- Las especificaciones se utilizan a menudo como una fuente adicional de información para determinar el resultado esperado de los casos de prueba

Las características comunes de las técnicas de prueba basadas en la experiencia incluyen las siguientes:

- Las condiciones de prueba, los casos de prueba y los datos de prueba se derivan de una base de prueba que puede incluir el conocimiento y la experiencia de los probadores, desarrolladores, usuarios y otras partes interesadas

Este conocimiento y experiencia incluye el uso esperado del software, su entorno, los posibles defectos y la distribución de dichos defectos.

La norma internacional (ISO/IEC/IEEE 29119-4) contiene descripciones de las técnicas de prueba y sus medidas de cobertura correspondientes (consulte Craig 2002 y Copeland 2004 para obtener más información sobre las técnicas).

4.2 Técnicas de prueba de caja negra

4.2.1 Segmentación de equivalencia

La segmentación de equivalencia divide los datos en segmentos (también conocidos como clases de equivalencia) de tal manera que se espera que todos los miembros de un segmento dado se procesen de la misma manera (consulte Kaner 2013 y Jorgensen 2014). Existen segmentaciones de equivalencia para valores válidos y no válidos.

- Los valores válidos son valores que deben ser aceptados por el componente o sistema.
- Una segmentación de equivalencia que contiene valores válidos se llama “segmentación de equivalencia válida”.
- Los valores no válidos son valores que deben ser rechazados por el componente o sistema.
- Las segmentaciones pueden identificarse para cualquier elemento de datos relacionado con el objeto de prueba, incluidas entradas, salidas, valores internos, valores relacionados con el tiempo (p. ej., antes o después de un evento) y para los parámetros de la interfaz (p. ej., componentes integrados que se están probando durante las pruebas de integración).
- Cualquier segmento se puede dividir en subsegmentos si es necesario.
- Cada valor debe pertenecer a una y solo una segmentación de equivalencia.
- Cuando se usan segmentaciones de equivalencia no válidas en los casos de prueba, deben probarse individualmente, es decir, no combinadas con otras segmentaciones de equivalencia no válidas, para garantizar que los fallos no estén enmascarados. Los fallos se pueden enmascarar cuando se producen varios fallos a la vez, pero solo uno es visible, lo que hace que los otros fallos no se detecten.

Para lograr una cobertura del 100% con esta técnica, los casos de prueba deben cubrir todas las segmentaciones identificadas (incluidas las segmentaciones no válidas) utilizando un mínimo de un valor de cada segmentación. La cobertura se mide como el número de segmentaciones de equivalencia probadas por al menos un valor, dividido por el número total de segmentaciones de equivalencia identificadas, normalmente expresadas como un porcentaje. La segmentación de equivalencia es aplicable en todos los niveles de prueba.

4.2.2 Análisis del valor límite

El análisis de valor de límite (AVL) es una extensión de la segmentación de equivalencia, pero solo se puede usar cuando la segmentación está ordenada, que consiste en datos numéricos o secuenciales. Los valores mínimo y máximo (o los valores primeros y últimos) de una segmentación son sus valores límite (Beizer 1990).

Por ejemplo, supongamos que un campo de entrada acepta un solo valor entero como entrada, usando un teclado para limitar las entradas de modo que las entradas no enteras sean imposibles. El rango válido es de 1 a 5, inclusivo. Por lo tanto, hay tres segmentaciones de equivalencia: no válida (demasiado baja); válida; no válida (demasiado alta). Para la segmentación de equivalencia válida, los valores de límite son 1 y 5. Para la segmentación no válida (demasiado alta), los valores de límite son 6 y 9. Para la segmentación no válida (demasiado baja), solo hay un valor de límite, 0, porque esta es una segmentación con un solo miembro.

En el ejemplo anterior, identificamos dos valores de límite por límite. El límite entre no válido (demasiado bajo) y válido da los valores de prueba 0 y 1. El límite entre válido y no válido (demasiado alto) da los valores de prueba 5 y 6. Algunas variaciones de esta técnica identifican tres valores de límite por límite: los valores antes, en, y justo por encima el límite. En el ejemplo anterior, utilizando valores de límite de tres puntos, los valores de prueba de límite inferior son 0, 1 y 2, y los valores de prueba de límite superior son 4, 5, y 6

(Jorgensen 2014).

El comportamiento en los límites de las segmentaciones de equivalencia es más probable que sea incorrecto que el comportamiento dentro de las segmentaciones. Es importante recordar que tanto los límites especificados como los implementados pueden desplazarse a posiciones por encima o por debajo de sus posiciones previstas, pueden omitirse por completo o pueden complementarse con límites adicionales no deseados. El análisis y las pruebas de valor de límite revelarán casi todos estos defectos al obligar al software a mostrar comportamientos desde una segmentación distinta a la que debe pertenecer el valor límite.

El análisis del valor límite se puede aplicar en todos los niveles de prueba. Esta técnica se usa generalmente para probar los requisitos que requieren un rango de números (incluidas fechas y horas). La cobertura de límites para una segmentación se mide como el número de valores de límite probados, dividido por el número total de valores de prueba de límite identificados, normalmente expresados como un porcentaje.

4.2.3 Prueba de la tabla de decisiones

Las técnicas de pruebas combinatorias son útiles para probar la implementación de los requisitos del sistema que especifican cómo las diferentes combinaciones de condiciones producen resultados diferentes. Un enfoque para tales pruebas es la prueba de la tabla de decisiones.

Las tablas de decisiones son una buena manera de registrar reglas de negocio complejas que un sistema debe implementar. Al crear tablas de decisiones, el probador identifica las condiciones (en ocasiones entradas) y las acciones resultantes (en ocasiones salidas) del sistema. Estas forman las filas de la tabla, generalmente con las condiciones en la parte superior y las acciones en la parte inferior. Cada columna corresponde a una regla de decisión que define una combinación única de condiciones que resulta en la ejecución de las acciones asociadas con esa regla. Los valores de las condiciones y acciones generalmente se muestran como valores booleanos (verdaderos o falsos) o valores discretos (p. ej., rojo, verde, azul), pero también pueden ser números o rangos de números. Estos diferentes tipos de condiciones y acciones pueden encontrarse juntos en la misma tabla.

La notación común en las tablas de decisión es la siguiente:

Para las condiciones:

- Y significa que la condición es verdadera (también se puede mostrar como T o 1)
- N significa que la condición es verdadera (también se puede mostrar como F o 0)
- — significa que el valor de la condición no importa (también se puede mostrar como N/A)

Para las acciones:

- X significa que la acción debería ocurrir (también se puede mostrar como Y o T o 1)
- En blanco significa que la acción no debe ocurrir (también puede aparecer como - o N o F o 0)

Una tabla de decisiones completa tiene suficientes columnas para cubrir cada combinación de condiciones. La tabla se puede contraer al eliminar columnas que contienen combinaciones de condiciones imposibles, columnas que contienen combinaciones de condiciones posibles, pero no factibles, y columnas que prueban combinaciones de condiciones que no afectan el resultado. Para obtener más información sobre cómo contraer las tablas de decisiones, consulte ISTQB Programa de Estudio – Nivel Avanzado – Analista de Pruebas.

El estándar de cobertura mínimo común para las pruebas de la tabla de decisiones es tener al menos un caso de prueba por regla de decisión en la tabla. Esto típicamente implica cubrir todas las combinaciones de condiciones. La cobertura se mide como el número de reglas de decisión probadas por al menos un caso de prueba, dividido por el número total de reglas de decisión, normalmente expresado como un porcentaje.

La fortaleza de las pruebas de la tabla de decisiones es que ayuda a identificar todas las combinaciones importantes de condiciones, algunas de las cuales podrían pasarse por alto. También ayuda a encontrar cualquier brecha en los requisitos Puede aplicarse a todas las situaciones en las que el comportamiento del software depende de una combinación de condiciones, en cualquier nivel de prueba.

4.2.4 Pruebas de transición de estado

Los componentes o sistemas pueden responder de manera diferente a un evento según las condiciones actuales o el historial anterior (p. ej., los eventos que han ocurrido desde que se inicializó el sistema). La historia anterior se puede resumir usando el concepto de estados. Un diagrama de transición de estado muestra los posibles estados del software, así como la forma en que el software ingresa, sale y hace transiciones entre los estados. Una transición se inicia mediante un evento (p. ej., la entrada del usuario de un valor en un campo). El evento resulta en una transición. Si el mismo evento puede resultar en dos o más transiciones diferentes del mismo estado, ese evento puede ser calificado por una condición de protección. El cambio de estado puede hacer que el software realice una acción (p. ej., emitir un cálculo o mensaje de error).

Una tabla de transición de estado muestra todas las transiciones válidas y las transiciones potencialmente no válidas entre estados, así como los eventos, las condiciones de protección y las acciones resultantes para las transiciones válidas. Los diagramas de transición de estado normalmente muestran solo las transiciones válidas y excluyen las transiciones no válidas.

Las pruebas pueden diseñarse para cubrir una secuencia típica de estados, para ejercitar todos los estados, para ejercer cada transición, para ejercer secuencias específicas de transiciones o para probar transiciones no válidas.

Las pruebas de transición de estado se utilizan para aplicaciones basadas en menús y se utilizan ampliamente en la industria del software integrado. La técnica también es adecuada para modelar un escenario empresarial con estados específicos o para probar la navegación en la pantalla. El concepto de estado es abstracto: puede representar unas pocas líneas de código o un proceso empresarial completo.

La cobertura se mide comúnmente como el número de estados identificados o transiciones probadas, dividido por el número total de estados identificados o transiciones en el objeto de prueba, normalmente expresado como un porcentaje. Para obtener más información sobre los criterios de cobertura para las pruebas de transición de estado, consulte ISTQB Programa de Estudio – Nivel Avanzado – Analista de Pruebas.

4.2.5 ruebas de caso de uso

Las pruebas pueden derivarse de casos de uso, que son una forma específica de diseñar interacciones con elementos de software, incorporando requisitos para las funciones de software representadas por los casos de uso. Los casos de uso están asociados con actores (usuarios humanos, hardware externo u otros componentes o sistemas) y temas (el componente o sistema al que se aplica el caso de uso).

Cada caso de uso especifica algún comportamiento que un sujeto puede realizar en colaboración con uno o más actores (UML 2.5.1 2017). Un caso de uso puede describirse mediante interacciones y actividades, así como condiciones previas, condiciones posteriores y lenguaje natural cuando sea apropiado. Las interacciones entre los actores y el sujeto pueden dar como resultado cambios en el estado del sujeto. Las interacciones pueden representarse gráficamente mediante flujos de trabajo, diagramas de actividad o modelos de procesos de negocio.

Un caso de uso puede incluir posibles variaciones de su comportamiento básico, incluido el comportamiento excepcional y el manejo de errores (respuesta del sistema y recuperación de errores de programación, aplicación y comunicación, p. ej., que resultan en un mensaje de error). Las pruebas están diseñadas para ejercer los comportamientos definidos (básico, excepcional o alternativo y manejo de errores). La cobertura se puede medir por el porcentaje de comportamientos de casos de uso probados dividido por el número total de comportamientos de casos de uso, normalmente expresados como un porcentaje.

Para obtener más información sobre los criterios de cobertura para las pruebas de caso, consulte ISTQB Programa de Estudio – Nivel Avanzado – Analista de Pruebas.

4.3 Técnicas de prueba de caja blanca

La prueba de caja blanca se basa en la estructura interna del objeto de prueba. Las técnicas de prueba de caja blanca se pueden usar en todos los niveles de prueba, pero las dos técnicas relacionadas con el código que se analizan en esta sección son las más utilizadas en el nivel de prueba de componentes. Existen técnicas más avanzadas que se utilizan en algunos entornos de seguridad crítica, de misión crítica o de alta integridad para lograr una cobertura más completa, pero no se tratan aquí. Para obtener más información sobre dichas técnicas, consulte ISTQB Programa de Estudio – Nivel Avanzado – Analista de Pruebas.

4.3.1 Pruebas de sentencia y cobertura

La prueba de sentencia ejerce las instrucciones ejecutables en el código. La cobertura se mide como el número de sentencias ejecutadas por las pruebas, dividido por el número total de sentencias ejecutables en el objeto de prueba, normalmente expresado como un porcentaje.

4.3.2 Pruebas de decisión y cobertura

Las pruebas de decisión ejercen las decisiones en el código y prueba el código que se ejecuta en función de los resultados de la decisión. Para hacer esto, los casos de prueba siguen los flujos de control que ocurren desde un punto de decisión (p. ej., para una declaración IF, uno para el resultado verdadero y otro para el resultado falso; para una sentencia CASE, se requerirán casos de prueba para todos los posibles resultados, incluido el resultado por defecto).

La cobertura se mide como el número de resultados de decisión ejecutados por las pruebas, dividido por el número total de resultados de decisión en el objeto de prueba, normalmente expresado como un porcentaje.

4.3.3 El valor de las pruebas de sentencia y de decisión

Cuando se alcanza el 100% de cobertura de estado de cuenta, se garantiza que todas las sentencias ejecutables en el código se hayan probado al menos una vez, pero no se garantiza que se haya probado toda la lógica de decisión. De las dos técnicas de caja blanca discutidas en este programa, la prueba de estado de cuenta puede proporcionar menos cobertura que las pruebas de decisión.

Cuando se alcanza el 100% de cobertura de decisión, ésta ejecuta todos los resultados de la decisión, lo que incluye probar el resultado verdadero y también el resultado falso, incluso cuando no hay una declaración falsa explícita (p. ej., en el caso de una sentencia IF sin otra cosa en el código). La cobertura de estados de cuenta ayuda a encontrar defectos en el código que no se aplicó en otras pruebas. La cobertura de decisiones ayuda a encontrar defectos en el código donde otras pruebas no han obtenido resultados verdaderos ni falsos.

Lograr una cobertura de decisión del 100% garantiza una cobertura del estado de cuenta del 100% (pero no al revés).

4.4 Técnicas de prueba basadas en la experiencia

Al aplicar técnicas de prueba basadas en la experiencia, los casos de prueba se derivan de la habilidad y la intuición del probador, y de su experiencia con aplicaciones y tecnologías similares. Estas técnicas pueden ser útiles para identificar pruebas que no se identificaron fácilmente con otras técnicas más sistemáticas. Dependiendo del enfoque y la experiencia del probador, estas técnicas pueden lograr grados de cobertura y efectividad muy variables. La cobertura puede ser difícil de evaluar y puede no ser medible con estas técnicas.

Las técnicas basadas en la experiencia usadas frecuentemente se analizan en las siguientes secciones.

4.4.1 Predicción de errores

La predicción de errores es una técnica utilizada para anticipar la ocurrencia de errores, defectos y fallos, según el conocimiento del probador, que incluye:

- Cómo ha funcionado la aplicación en el pasado
- Qué tipo de errores tienden a cometer los desarrolladores
- Fallos que han ocurrido en otras aplicaciones

Un enfoque metódico de la técnica de predicción de errores es crear una lista de posibles errores, defectos y fallos, y diseñar pruebas que expondrán esos fallos y los defectos que los causaron. Estas listas de errores, defectos, fallos pueden construirse en base a la experiencia, datos de fallos y defectos, o de conocimiento común acerca de por qué falla el software.

4.4.2 Pruebas exploratorias

En las pruebas exploratorias, las pruebas informales (no predefinidas) se diseñan, ejecutan, registran y evalúan dinámicamente durante la ejecución de la prueba. Los resultados de las pruebas se utilizan para aprender más sobre el componente o sistema, y para crear pruebas para las áreas que pueden necesitar más pruebas.

Las pruebas exploratorias a menudo se llevan a cabo utilizando pruebas basadas en sesiones para estructurar la actividad. En las pruebas basadas en sesiones, las pruebas exploratorias se llevan a cabo dentro de un cuadro de tiempo definido, y el probador utiliza un contrato de la prueba que contiene los objetivos de la prueba para guiar la misma. El probador puede usar las hojas de sesión de prueba para documentar los pasos seguidos y los descubrimientos realizados.

Las pruebas exploratorias son más útiles cuando hay pocas especificaciones o inadecuadas o una presión de tiempo significativa en las pruebas. Las pruebas exploratorias también son útiles para complementar otras técnicas de prueba más formales.

Las pruebas exploratorias están fuertemente asociadas con las estrategias de pruebas reactivas (ver sección 5.2.2). Las pruebas exploratorias pueden incorporar el uso de otras técnicas basadas en la caja negra, la caja blanca yen la experiencia.

4.4.3 Pruebas basadas en listas de comprobación

En las pruebas basadas en listas de comprobación, los probadores diseñan, implementan y ejecutan pruebas para cubrir las condiciones de prueba encontradas en una lista de comprobación. Como parte del análisis, los evaluadores crean una nueva lista de comprobación o expanden una lista de comprobación existente, pero los probadores también pueden usar una lista de comprobación existente sin modificaciones. Estas listas de comprobación pueden construirse en base a la experiencia, el conocimiento sobre lo que es importante para el usuario o un entendimiento de por qué y cómo falla el software.

Se pueden crear listas de comprobación para admitir varios tipos de pruebas, incluidas las pruebas funcionales y no funcionales. En ausencia de casos de prueba detallados, las pruebas basadas en listas de comprobación pueden proporcionar pautas y un cierto grado de coherencia. Como se trata de listas de alto nivel, es probable que se produzca cierta variabilidad en las pruebas reales, lo que puede dar como resultado una mayor cobertura pero una menor repetibilidad.

5 Gestión de Prueba — 225 minutos

Palabras clave

gestión de configuración, gestión de defectos, criterios de entrada, criterios de salida, riesgo del producto, riesgo del proyecto, riesgo, nivel de riesgo, pruebas basadas en el riesgo, enfoque de prueba, control de prueba, estimación de la prueba, jefe de prueba, monitorización de la prueba, plan de prueba, planificación de prueba, informe del avance de la prueba, estrategia de prueba, informe resumen de prueba, probador

Objetivos de Aprendizaje para la gestión de prueba

5.1 Organización de la prueba

FL-5.1.1 (K2) Explicar los beneficios e inconvenientes de las pruebas independientes

FL-5.1.2 (K1) Identificar las tareas de un jefe de prueba y de un probador

5.2 Planificación y estimación de la prueba

FL-5.2.1 (K2) Resumir el propósito y el contenido de un plan de prueba

FL-5.2.2 (K2) Diferenciar entre varias estrategias de prueba

FL-5.2.3 (K2) Dar ejemplos de posibles criterios de entrada y salida

FL-5.2.4 (K3) Aplicar el conocimiento de la priorización y las dependencias técnicas y lógicas para programar la ejecución de la prueba para un conjunto dado de casos de prueba

FL-5.2.5 (K1) Identificar los factores típicos que influyen en el esfuerzo relacionado con las pruebas

FL-5.2.6 (K2) Explicar la diferencia entre dos técnicas de estimación: la técnica basada en métricas y la técnica basada en expertos

5.3 Monitorización y control de la prueba

FL-5.3.1 (K1) Recordar las métricas utilizadas para las pruebas

FL-5.3.2 (K2) Resumir los propósitos, contenidos y destinatarios de los informes de prueba

5.4 Gestión de configuración

FL-5.4.1 (K2) Resumir cómo la gestión de configuración apoya las pruebas

5.5 Los Riesgos y las Pruebas

FL-5.5.1 (K1) Definir el nivel de riesgo utilizando la probabilidad y el impacto

FL-5.5.2 (K2) Distinguir entre los riesgos del proyecto y del producto

FL-5.5.3 (K2) Describir, usando ejemplos, cómo el análisis de riesgo del producto puede influir en la exhaustividad y el alcance de las pruebas

5.6 Gestión de defectos

FL-5.6.1 (K3) Redactar un informe de defectos, que cubra los defectos encontrados durante la prueba

5.1 Organización de la prueba

5.1.1 Pruebas independientes

Las tareas de prueba pueden ser realizadas por personas en un rol de prueba específico o por personas en otro rol (p. ej., clientes). Un cierto grado de independencia a menudo hace que el probador sea más eficaz para detectar defectos debidos a diferencias entre los sesgos cognitivos del autor y del probador (ver sección 1.5). Sin embargo, la independencia no es un reemplazo para la familiaridad, y los desarrolladores pueden encontrar muchos defectos en su propio código de manera eficiente.

Los grados de independencia en las pruebas incluyen lo siguiente (desde un nivel bajo de independencia hasta un nivel alto):

- Sin probadores independientes; la única forma de prueba disponible es que los desarrolladores prueben su propio código
- Desarrolladores o probadores independientes dentro de los equipos de desarrollo o el equipo del proyecto; podría ser desarrolladores que prueben los productos de sus colegas
- Equipo o grupo de prueba independiente dentro de la organización, que informa a la administración del proyecto o a la dirección administrativa
- Probadores independientes de la organización empresarial o comunidad de usuarios, o con especializaciones en tipos de pruebas específicos, tales como facilidad de uso, seguridad, rendimiento, regulación/cumplimiento o portabilidad
- Probadores independientes ajenos a la organización, ya sea trabajando en el sitio (dentro de la compañía) o fuera del sitio (subcontratación externa)

Para la mayoría de los tipos de proyectos, generalmente es mejor tener múltiples niveles de prueba, con algunos de estos niveles manejados por probadores independientes. Los desarrolladores deben participar en las pruebas, especialmente en los niveles inferiores, a fin de ejercer control sobre la calidad de su propio trabajo.

La forma en que se implementa la independencia de las pruebas varía según el modelo de ciclo de vida del desarrollo de software. Por ejemplo, en el desarrollo Ágil, los probadores pueden formar parte de un equipo de desarrollo. n algunas organizaciones que usan métodos Ágiles, estos probadores también pueden considerarse parte de un equipo de pruebas independiente más grande. Además, en dichas organizaciones, los propietarios de productos pueden realizar pruebas de aceptación para validar las historias de usuario al final de cada iteración.

Las ventajas potenciales de la independencia de la prueba incluyen:

- Los probadores independientes pueden reconocer diferentes tipos de fallos en comparación con los desarrolladores debido a sus diferentes trayectorias, perspectivas técnicas y sesgos
- Un probador independiente puede verificar, desafiar o refutar las premisas hechas por las partes interesadas durante la especificación e implementación del sistema

Los posibles inconvenientes de la independencia de la prueba incluyen:

- Aislamiento del equipo de desarrollo, lo que lleva a una falta de colaboración, retrasos en la retroalimentación al equipo de desarrollo o una relación adversa con el equipo de desarrollo
- Los desarrolladores pueden perder el sentido de responsabilidad por la calidad
- Los probadores independientes pueden verse como un cuello de botella o ser culpados por retrasos en la entrega

- Los probadores independientes pueden carecer de información importante (p. ej., sobre el objeto de prueba)

Muchas organizaciones pueden lograr con éxito los beneficios de la independencia de las pruebas, evitando los inconvenientes.

5.1.2 Tareas de un jefe de prueba y de un probador

En este programa de estudio se cubren dos roles de prueba, los jefes de prueba y los probadores. Las actividades y tareas realizadas por estos dos roles dependen del contexto del proyecto y del producto, las habilidades de las personas en los roles y la organización.

El jefe de prueba tiene la responsabilidad general del proceso de prueba y el liderazgo exitoso de las actividades de prueba La función de gestión de pruebas puede ser realizada por un jefe de prueba profesional, o por un jefe de proyecto, un director de desarrollo o un director de control de calidad. En proyectos u organizaciones más grandes, varios equipos de pruebas pueden informar a un jefe de pruebas, entrenador de pruebas o coordinador de pruebas, y cada equipo está encabezado por un líder de la prueba o un probador principal.

Las tareas típicas del jefe de prueba pueden incluir:

- Desarrollar o revisar una política de prueba y una estrategia de prueba para la organización
- Planificar las actividades de la prueba considerando el contexto y entendiendo los objetivos y riesgos de la prueba. Esto puede incluir la selección de enfoques de prueba, la estimación del tiempo, el esfuerzo y el costo de la prueba, la adquisición de recursos, la definición de los niveles de prueba y los ciclos de prueba, y la planificación de la gestión de defectos
- Escribir y actualizar los planes de prueba
- Coordinar el (los) plan (es) de prueba con los jefes de proyecto, propietarios de productos y otros
- Compartir las perspectivas de prueba con otras actividades del proyecto, como la planificación de la integración
- Iniciar el análisis, diseño, implementación y ejecución de pruebas, monitorear el progreso de la prueba y los resultados, y verificar el estado de los criterios de salida (o la definición de hecho)
- Preparar y entregar informes del avance de la prueba e informes resúmenes de pruebas basados en la información recopilada
- Adaptar la planificación basada en los resultados de las pruebas y el avance (a veces documentado en los informes de avance de las pruebas y/o en los informes resúmenes de pruebas para otras pruebas ya completadas en el proyecto) y tomar las medidas necesarias para el control de las pruebas
- Soporte para configurar el sistema de gestión de defectos y la gestión de configuración adecuada de productos de prueba
- Introducir métricas adecuadas para medir el avance de las pruebas y evaluar la calidad de las pruebas y el producto
- Apoyar la selección e implementación de herramientas para respaldar el proceso de prueba, incluida la recomendación del presupuesto para la selección de la herramienta (y posiblemente la compra y/o el soporte), la asignación de tiempo y esfuerzo para los proyectos pilotos y el apoyo continuo en el uso de la(s) herramienta(s)
- Decidir sobre la implementación de (los) entorno (s) de prueba
- Promover y abogar por los probadores, el equipo de prueba y la profesión de prueba dentro de la organización
- Desarrollar las habilidades y carreras de los probadores (p. ej., a través de planes de capacitación,

evaluaciones de desempeño, entrenamiento, etc.)

La forma en que se lleva a cabo la función de jefe de pruebas varía según el ciclo de vida del desarrollo del software. Por ejemplo, en el desarrollo Ágil, algunas de las tareas mencionadas anteriormente son manejadas por el equipo Ágil, especialmente aquellas tareas relacionadas con las pruebas diarias realizadas dentro del equipo, a menudo por un probador que trabaja dentro del equipo. Algunas de las tareas que abarcan varios equipos o toda la organización, o que tienen que ver con la administración de personal, pueden realizarlos los jefes de prueba fuera del equipo de desarrollo, que a veces se llaman instructores de pruebas. Consulte a Black 2009 para más información sobre la gestión del proceso de prueba.

Las tareas típicas del probador pueden incluir:

- Revisar y aportar a los planes de prueba
- Analizar, revisar y evaluar los requisitos, las historias de usuario y los criterios de aceptación, las especificaciones y los modelos para la prueba (es decir, la base de prueba)
- Identificar y documentar las condiciones de prueba, y capturar la trazabilidad entre los casos de prueba, las condiciones de prueba y la base de prueba
- Diseñar, configurar y verificar entornos de prueba, a menudo coordinando con la administración del sistema y la administración de la red
- Diseñar e implementar casos de prueba y procedimientos de prueba
- Preparar y adquirir datos de prueba
- Crear el programa detallado de ejecución de la prueba
- Ejecutar pruebas, evaluar los resultados y documentar las desviaciones de los resultados esperados
- Usar herramientas apropiadas para facilitar el proceso de prueba
- Automatizar las pruebas según sea necesario (puede ser apoyado por un desarrollador o un experto en automatización de pruebas)
- Evaluar las características de calidad no funcionales, como son la eficiencia del rendimiento, fiabilidad, seguridad, compatibilidad y portabilidad
- Revisar pruebas desarrolladas por otros

Las personas que trabajan en análisis de pruebas, diseño de pruebas, tipos de pruebas específicos o automatización de pruebas pueden ser especialistas en estas funciones. Dependiendo de los riesgos relacionados con el producto y el proyecto, y del modelo de ciclo de vida de desarrollo de software seleccionado, diferentes personas pueden asumir el rol de probador en diferentes niveles de prueba. Por ejemplo, en el nivel de prueba de componentes y el nivel de prueba de integración de componentes, el rol de un probador a menudo lo realizan los desarrolladores. En el nivel de prueba de aceptación, el rol de un probador a menudo lo realizan analistas de negocios, expertos en la materia y usuarios. En el nivel de prueba del sistema y el nivel de prueba de integración del sistema, la función de un probador a menudo la realiza un equipo de prueba independiente. En el nivel de prueba de aceptación operacional, el rol de un probador a menudo lo realizan el personal operaciones y/o de administración de sistemas.

5.2 Planificación y estimación de las pruebas

5.2.1 Propósito y contenido de un plan de pruebas

Un plan de prueba describe las actividades de prueba para proyectos de desarrollo y mantenimiento. La planificación está influenciada por la política de prueba y la estrategia de prueba de la organización, los ciclos de vida y los métodos de desarrollo que se utilizan (consulte la sección 2.1), el alcance de las pruebas, los objetivos, los riesgos, las limitaciones, la criticidad, la capacidad de ser probado y la disponibilidad de recursos.

A medida que avanza la planificación del proyecto y la prueba, se dispone de más información y se pueden incluir más detalles en el plan de prueba. La planificación de pruebas es una actividad continua y se realiza a lo largo del ciclo de vida del producto. (Tenga en cuenta que el ciclo de vida del producto puede extenderse más allá del alcance de un proyecto para incluir la fase de mantenimiento). La retroalimentación de las actividades de prueba se debe utilizar para reconocer los riesgos cambiantes para que la planificación se pueda ajustar. La planificación se puede documentar en un plan de prueba maestro y en planes de prueba separados para niveles de prueba, tales como pruebas de sistema y pruebas de aceptación, o para tipos de pruebas separadas, como son las pruebas de usabilidad y pruebas de rendimiento. Las actividades de planificación de pruebas pueden incluir lo siguiente y algunas de estas pueden documentarse en un plan de prueba:

- Determinar el alcance, objetivos y riesgos de las pruebas
- Definir el enfoque global de las pruebas
- Integrar y coordinar las actividades de prueba en las actividades del ciclo de vida del software
- Tomar decisiones sobre qué probar, las personas y otros recursos necesarios para realizar las diversas actividades de prueba, y cómo se llevarán a cabo las actividades de prueba
- Programar los análisis de prueba, diseño, implementación, ejecución y actividades de evaluación, ya sea en fechas específicas (p. ej., en el desarrollo secuencial) o en el contexto de cada iteración (p. ej., en el desarrollo iterativo)
- Seleccionar las métricas para monitorización y control de pruebas
- Presupuesto para las actividades de prueba
- Determinar el nivel de detalle y la estructura de la documentación de prueba (p. ej., proporcionando plantillas o documentos de ejemplo)

El contenido de los planes de prueba varía y puede extenderse más allá de los temas identificados anteriormente. Los planes de prueba de muestra se pueden encontrar en la norma ISO (ISO/IEC/IEEE 29119-3).

5.2.2 Estrategia de prueba y enfoque de prueba

Una estrategia de prueba proporciona una descripción general del proceso de prueba, generalmente a nivel de producto u organización. Los tipos comunes de estrategias de prueba incluyen:

- **Analítica:** Este tipo de estrategia de prueba se basa en un análisis de algún factor (p. ej., requisitos or riesgos). Las pruebas basadas en el riesgo son un ejemplo de un enfoque analítico, donde las pruebas se diseñan y priorizan según el nivel de riesgo.
- **Basadas en modelos:** En este tipo de estrategia de prueba, las pruebas se diseñan en función de algún modelo de algún aspecto requerido del producto, como una función, un proceso de negocios, una estructura interna o una característica no funcional (p. ej., confiabilidad). Los ejemplos de dichos modelos incluyen modelos de procesos de negocio, modelos de estado y modelos de crecimiento de confiabilidad.

- **Metódica** Este tipo de estrategia de prueba se basa en hacer un uso sistemático de un conjunto predefinido de pruebas o condiciones de prueba, como una taxonomía de tipos comunes o probables de fallas, una lista de características de calidad importantes, o el aspecto y normas de aspecto visual y operacional para aplicaciones móviles o páginas Web para toda la empresa.
- **Compatibles con los procesos** (o compatibles con las normas): Este tipo de estrategia de prueba implica analizar, diseñar e implementar pruebas basadas en reglas y normas externas, como las especificadas por normas específicas de la industria, por documentación del proceso, mediante la identificación rigurosa y uso de la base de prueba, o por cualquier proceso o norma impuesta a o por la organización.
- **Dirigidas** (o consultivas): Este tipo de estrategia de prueba se basa principalmente en el asesoramiento, la orientación o las instrucciones de las partes interesadas, los expertos en el dominio de negocios o los expertos en tecnología, que pueden estar fuera del equipo de prueba o fuera de la organización.
- **Adversas a la regresión:** Este tipo de estrategia de prueba está motivada por el deseo de evitar la regresión de las capacidades existentes. Esta estrategia de prueba incluye la reutilización del software de prueba existente (especialmente los casos de prueba y los datos de prueba), la automatización extensiva de las pruebas de regresión y los juegos de prueba estándar.
- **Reactivas:** En este tipo de estrategia de prueba, la prueba es reactiva al componente o sistema que se está probando, y los eventos que ocurren durante la ejecución de la prueba, en lugar de ser planificados previamente (como lo son las estrategias anteriores). Las pruebas se diseñan e implementan, y pueden ejecutarse inmediatamente en respuesta al conocimiento obtenido de los resultados de pruebas anteriores. La prueba exploratoria es una técnica común empleada en estrategias reactivas.

Una estrategia de prueba adecuada se crea a menudo combinando varios de estos tipos de estrategias de prueba. Por ejemplo, las pruebas basadas en el riesgo (una estrategia analítica) se pueden combinar con las pruebas exploratorias (una estrategia reactiva); se complementan entre sí y pueden lograr pruebas más efectivas cuando se usan juntas.

Si bien la estrategia de prueba proporciona una descripción general del proceso de prueba, el enfoque de prueba adapta la estrategia de prueba para un proyecto o entrega en particular. El enfoque de prueba es el punto de partida para seleccionar las técnicas de prueba, los niveles de prueba y los tipos de prueba, y para definir los criterios de entrada y los criterios de salida (o la definición de listo y definición de hecho, respectivamente). La adaptación de la estrategia se basa en las decisiones tomadas en relación con la complejidad y los objetivos del proyecto, el tipo de producto que se está desarrollando y el análisis del riesgo del producto. El enfoque seleccionado depende del contexto y puede considerar factores como riesgos, seguridad, recursos y habilidades disponibles, tecnología, la naturaleza del sistema (p. ej., construcción personalizada versus COTS), objetivos de prueba y regulaciones.

5.2.3 Criterios de entrada y criterios de salida (Definición de Listo y Definición de Hecho)

Para ejercer un control efectivo sobre la calidad del software y de las pruebas, es recomendable tener criterios que definan cuándo debe comenzar una determinada actividad de prueba y cuándo se completa la actividad. Los criterios de entrada (más comúnmente denominados definiciones de listo en el desarrollo Ágil) definen las condiciones previas para llevar a cabo una actividad de prueba determinada. Si no se cumplen los criterios de entrada, es probable que la actividad resulte más difícil, más lenta, más costosa y más riesgosa. Los criterios de salida (más comúnmente llamados definición de hecho en el desarrollo Ágil) definen qué condiciones deben lograrse para declarar un nivel de prueba o un juego de pruebas completadas. Los criterios de entrada y salida deben definirse para cada nivel de prueba y tipo de prueba, y diferirán según los objetivos de prueba.

Los criterios típicos de entrada incluyen:

- Disponibilidad de requisitos susceptibles de ser probados, historias de usuario y/o modelos (p. ej., al seguir una estrategia de prueba basada en modelos)
- Disponibilidad de elementos de prueba que han cumplido con los criterios de salida para cualquier nivel de prueba anterior
- Disponibilidad del entorno de prueba
- Disponibilidad de herramientas de prueba necesarias
- Disponibilidad de datos de prueba y otros recursos necesarios

Los criterios típicos de salida incluyen:

- Las pruebas previstas han sido ejecutadas
- Se ha alcanzado un nivel definido de cobertura (p. ej., de requisitos, historias de usuario, criterios de aceptación, riesgos, Código)
- El número de defectos no resueltos está dentro de un límite acordado
- El número estimado de defectos restantes es suficientemente bajo
- Los niveles evaluados de confiabilidad, eficiencia de rendimiento, usabilidad, seguridad y otras características de calidad relevantes son suficientes

Incluso si no se satisfacen los criterios de salida, también es común que las actividades de prueba se reduzcan debido al gasto del presupuesto, a la finalización del tiempo programado y a la presión para llevar el producto al mercado. Puede ser aceptable finalizar las pruebas en tales circunstancias, si las partes interesadas del proyecto y los propietarios de las empresas han revisado y aceptado el riesgo de puesta en marcha sin más pruebas.

5.2.4 Programa de ejecución de la prueba

Una vez que los diversos casos de prueba y los procedimientos de prueba se producen (con algunos procedimientos de prueba potencialmente automatizados) y se ensamblan en juegos de prueba, los juegos de prueba pueden organizarse en un programa de ejecución de prueba que define el orden en el que se ejecutarán. El cronograma de ejecución de la prueba debe tener en cuenta factores como la priorización, las dependencias, las pruebas de confirmación, las pruebas de regresión y la secuencia más eficiente para ejecutar las pruebas.

Lo ideal es que los casos de prueba se ordenen ejecutar en función de sus niveles de prioridad, por lo general, ejecutando los casos de prueba con la prioridad más alta primero. Sin embargo, esta práctica puede no funcionar si los casos de prueba tienen dependencias o las funciones que se están probando tienen dependencias. Si un caso de prueba con una prioridad más alta depende de un caso de prueba con una prioridad más baja, el caso de prueba de prioridad más baja debe ejecutarse primero.
De manera similar, si hay dependencias entre los casos de prueba, deben ordenarse de manera adecuada independientemente de sus prioridades relativas. Las pruebas de confirmación y regresión también deben priorizarse, según la importancia de una retroalimentación rápida sobre los cambios, pero aquí nuevamente pueden aplicarse las dependencias.

En algunos casos, varias secuencias de pruebas son posibles, con diferentes niveles de eficiencia asociados con esas secuencias En tales casos, se deben hacer concesiones entre la eficiencia de la ejecución de la prueba y el cumplimiento de la priorización.

5.2.5 Factores que influyen en el esfuerzo de prueba

La estimación del esfuerzo de prueba implica predecir la cantidad de trabajo relacionado con la prueba que se necesitará para cumplir los objetivos de prueba para un proyecto, entrega o iteración en particular. Los factores que influyen en el esfuerzo de prueba pueden incluir las características del producto, las características del proceso de desarrollo, las características de las personas y los resultados de la prueba, como se muestra a continuación.

Características del producto

- Los riesgos asociados al producto
- La calidad de la base de prueba
- El tamaño del producto
- La complejidad del dominio del producto
- Los requisitos para las características de calidad (p. ej., seguridad, confiabilidad)
- El nivel de detalle requerido para la documentación de la prueba
- Requisitos para el cumplimiento de la legalidad y de las regulaciones

Características del proceso de desarrollo

- La estabilidad y madurez de la organización
- El modelo de desarrollo en uso
- El enfoque de prueba
- Las herramientas utilizadas
- El proceso de prueba
- La presión del tiempo

Características de las personas

- Las habilidades y la experiencia de las personas involucradas, especialmente con proyectos y productos similares (p. ej., conocimiento del dominio)
- Cohesión y liderazgo del equipo

Resultados de la prueba

- El número y severidad de los defectos encontrados
- La cantidad de restauración requerida

5.2.6 Técnicas de estimación de prueba

Hay una serie de técnicas de estimación utilizadas para determinar el esfuerzo requerido para realizar pruebas adecuadas. Dos de las técnicas más utilizadas son:

- La técnica basada en métricas: Estimar el esfuerzo de prueba basado en métricas de proyectos similares anteriores, o en valores típicos
- La técnica basada en expertos: La estimación del esfuerzo de prueba en base a la experiencia de los propietarios de las tareas de prueba o de expertos

Por ejemplo, en el desarrollo Ágil, los diagramas de trabajo pendiente son ejemplos del enfoque basado en métricas a medida que se captura e informa el esfuerzo, y luego se utiliza para alimentar la velocidad del equipo para determinar la cantidad de trabajo que el equipo puede hacer en la próxima iteración; mientras que la planificación póker es un ejemplo del enfoque basado en expertos, ya que los miembros del equipo están estimando el esfuerzo para ofrecer una característica basada en su experiencia (ISTQB-AT Extensión del Programa de Estudio – Nivel Básico – Probador Ágil).

Dentro de los proyectos secuenciales, los modelos de eliminación de defectos son ejemplos del enfoque basado en métricas, donde se capturan y reportan los volúmenes de defectos y el tiempo para eliminarlos, que luego proporciona una base para estimar proyectos futuros de naturaleza similar; mientras que la técnica de estimación Wideband Delphi es un ejemplo del enfoque basado en expertos en el que los grupos de

expertos proporciona estimaciones basadas en su experiencia (ISTQB Programa de Estudio – Nivel Avanzado – Jefe de Pruebas).

5.3 Monitorización y Control de la Prueba

El propósito de la monitorización de pruebas es recopilar información y brindar retroalimentación y visibilidad sobre las actividades de prueba. La información que se debe monitorear se puede recopilar de forma manual o automática y se debe utilizar para evaluar el avance de la prueba y para medir si se cumplen los criterios de salida de la prueba o las tareas de prueba asociadas con la definición de hecho de un proyecto Ágil, tales como cumplir con los objetivos de cobertura de riesgos del producto, requisitos, o criterios de aceptación.

El control de prueba describe cualquier guía o acción correctiva tomada como resultado de la información y las métricas recopiladas y (posiblemente) informadas. Las acciones pueden cubrir cualquier actividad de prueba y pueden afectar cualquier otra actividad del ciclo de vida del software.

Ejemplos de acciones de control de prueba incluyen:

- Volver a priorizar las pruebas cuando se produce un riesgo identificado (p. ej., el software se entrega con retraso)
- Cambiar el programa de pruebas debido a la disponibilidad o falta de disponibilidad de un entorno de prueba u otros recursos
- Reevaluar si un elemento de prueba cumple con un criterio de entrada o salida debido a un nuevo trabajo

5.3.1 Métrica utilizada en las pruebas

Las métricas se pueden recopilar durante y al final de las actividades de prueba que se van a evaluar:

- Avance contra el cronograma y presupuesto planificados
- Calidad actual del objeto de prueba
- Adecuación del enfoque de prueba
- Eficacia de las actividades de prueba con respecto a los objetivos

Las métricas de prueba comunes incluyen:

- Porcentaje de trabajo planificado realizado en la preparación de casos de prueba (o porcentaje de casos de prueba planificados implementados)
- Porcentaje del trabajo planificado realizado en la preparación del entorno de prueba
- Ejecución de casos de prueba (p. ej., número de casos de prueba ejecutados/no ejecutados, casos de prueba que pasan/fallan y/o condiciones de prueba que pasan/fallan)
- Información de defectos (p. ej., densidad de defectos, defectos encontrados y corregidos, tasa de fallos y resultados de las pruebas de confirmación)
- Cobertura de prueba de requisitos, historias de usuario, criterios de aceptación, riesgos o código
- Finalización de tareas, asignación y uso de recursos, y esfuerzo
- El costo de la prueba, incluido el costo comparado con el beneficio de encontrar el próximo defecto o el costo comparado con el beneficio de ejecutar la siguiente prueba

5.3.2 Propósitos, contenidos y audiencias para la gestión de información de la prueba

El propósito de la gestión de información de la prueba es resumir y comunicar la información de la actividad de prueba, tanto durante como al final de una actividad de prueba (p. ej., un nivel de prueba). La información de la prueba preparada durante una actividad de prueba puede referirse como un informe del avance de la prueba, mientras que la información de la prueba preparada al final de una actividad de prueba puede referirse como un informe resumen de prueba.

Durante la monitorización y control de las pruebas, el jefe de pruebas emite regularmente informes del avance de la prueba para las partes interesadas. Además del contenido común para los informes del avance de la prueba y los informes resumen de prueba, los informes del avance de la prueba típicos también pueden incluir:

- El estado de las actividades de prueba y el avance en comparación con el plan de prueba
- Factores que impiden el avance
- Pruebas programadas para el próximo periodo de información
- La calidad del objeto de prueba

Cuando se alcanzan los criterios de salida, el jefe de prueba emite el informe de resumen de prueba. Este informe proporciona un resumen de las pruebas realizadas, según el último informe del avance de las pruebas y cualquier otra información relevante.

Los informes típicos de avance de la prueba y los informes resumen de prueba pueden incluir:

- Resumen de las pruebas realizadas
- Información sobre lo que ocurrió durante un período de prueba
- Desviaciones del plan, incluidas las desviaciones en el cronograma, la duración o el esfuerzo de las actividades de prueba
- Estado de prueba y calidad del producto con respecto a los criterios de salida o definición de hecho
- Factores que han bloqueado o continúan bloqueando el avance
- Métricas de defectos, casos de prueba, cobertura de prueba, avance de la actividad y consumo de recursos (p. ej., como se describe en 5.3.1)
- Riesgos residuales (ver sección 5.5)
- Productos de trabajo de prueba reutilizables producidos

El contenido de un informe de prueba variará según el proyecto, los requisitos de la organización y el ciclo de vida del desarrollo del software. Por ejemplo, un proyecto complejo con muchas partes interesadas o un proyecto regulado puede requerir informes más detallados y rigurosos que una actualización rápida de software. Como otro ejemplo, en el desarrollo Ágil, el informe del avance de la prueba se puede incorporar en los tabla s de tareas, los resúmenes de defectos y los diagramas de trabajo pendiente, que se pueden discutir durante una reunión de pie diaria (consulte ISTQB-AT Extensión del Programa de Estudio – Nivel Básico – Probador Ágil).

Además de adaptar los informes de prueba según el contexto del proyecto, los informes de prueba deben adaptarse a la audiencia del informe. El tipo y la cantidad de información que se debe incluir para una audiencia técnica o un equipo de prueba puede ser diferente de lo que se incluiría en un informe de resumen ejecutivo. En el primer caso, la información detallada sobre los tipos de defectos y las tendencias puede ser importante. En el último caso, un informe de alto nivel (p. ej., un resumen de estado de los defectos por prioridad, presupuesto, calendario y condiciones de prueba que pasan/fallan/no probadas) puede ser más apropiado.

La norma ISO (ISO/IEC/IEEE 29119-3) se refiere a dos tipos de informes de prueba, informes de avance de prueba e informes de finalización de prueba (llamados informes resumen de prueba en este programa), y contiene estructuras y ejemplos para cada tipo.

5.4 Gestión de la Configuración

El propósito de la gestión de la configuración es establecer y mantener la integridad del componente o sistema, el software de prueba y sus relaciones entre sí a través del ciclo de vida del proyecto y del producto.

Para apoyar adecuadamente las pruebas, la gestión de la configuración puede implicar asegurar lo siguiente:

- Todos los elementos de prueba se identifican de forma única, se controlan las versiones, se rastrean los cambios y se relacionan entre sí.
- Todos los elementos del software de prueba se identifican de forma única, se controlan las versiones, se rastrean los cambios, se relacionan entre sí y se relacionan con las versiones de los elementos de prueba, de modo que la trazabilidad se pueda mantener durante todo el proceso de prueba.
- Todos los documentos y elementos de software identificados se mencionan de forma inequívoca en la documentación de prueba.

Durante la planificación de prueba, se deben identificar e implementar los procedimientos de gestión de configuración y la infraestructura (herramientas).

5.5 Los Riesgos y las Pruebas

5.5.1 Definición de riesgo

El riesgo implica la posibilidad de un evento en el futuro que tenga consecuencias negativas. El nivel de riesgo está determinado por la probabilidad del evento y el impacto (el daño) de ese evento.

5.5.2 Riesgos del producto y del proyecto

El riesgo del producto implica la posibilidad de que un producto de trabajo (p. ej., una especificación, componente, sistema o prueba) no logre satisfacer las necesidades legítimas de sus usuarios y/o partes interesadas. Cuando los riesgos del producto se asocian con características de calidad específicas de un producto (p. ej., idoneidad funcional, confiabilidad, eficiencia de rendimiento, facilidad de uso, seguridad, compatibilidad, mantenibilidad y portabilidad), los riesgos del producto también se denominan riesgos de calidad. Los ejemplos de riesgos de productos incluyen:

- El software pudiera no realizar sus funciones previstas de acuerdo con la especificación
- Es posible que el software no realice las funciones previstas de acuerdo con las necesidades del usuario, cliente y/o partes interesadas
- Es posible que una arquitectura del sistema no admita adecuadamente algunos requisitos no funcionales
- Un cálculo específico pudiera realizarse incorrectamente en algunas circunstancias
- Una estructura de control de bucle pudiera estar codificada incorrectamente
- Los tiempos de respuesta pueden ser inadecuados para un sistema de procesamiento de transacciones de alto rendimiento
- Es posible que los comentarios de la experiencia del usuario (UX) no cumplan con las expectativas del producto

El riesgo del proyecto implica situaciones que, en caso de que ocurran, pueden tener un efecto negativo sobre la capacidad de un proyecto para lograr sus objetivos. Los ejemplos de riesgos del proyecto incluyen:

- Problemas del proyecto:
  - Pueden producirse retrasos en la entrega, la finalización de la tarea o la satisfacción de los criterios de salida o la definición de hecho
  - Estimaciones inexactas, la reasignación de fondos a proyectos de mayor prioridad o el recorte general de costos en toda la organización puede resultar en una financiación inadecuada
  - Los cambios tardíos pueden dar lugar a una restauración sustancial
- Problemas de la organización:
  - Las habilidades, capacitación y personal pueden no ser suficientes
  - Los problemas de personal pueden causar conflictos y problemas
  - Es posible que los usuarios, el personal de negocios o los expertos en la materia no estén disponibles debido a prioridades comerciales conflictivas
- Problemas de políticas:
  - Los probadores no pueden comunicar sus necesidades y/o los resultados de las pruebas de manera adecuada
  - Los desarrolladores y/o probadores pueden fallar en el seguimiento de la información que se encuentra en las pruebas y revisiones (p. ej., no mejora las prácticas de desarrollo y de pruebas)
  - Puede haber una actitud inapropiada hacia las pruebas, o expectativas, (p. ej., no apreciar el valor de encontrar defectos durante las pruebas)
- Problemas técnicos:
  - Los requisitos pueden no estar suficientemente definidos
  - Los requisitos pueden no cumplirse, dadas las limitaciones existentes
  - El entorno de prueba puede no estar listo a tiempo
  - La conversión de datos, la planificación de la migración y su soporte de herramientas pueden llegar tarde
  - Las debilidades en el proceso de desarrollo pueden afectar la consistencia o la calidad de los productos de trabajo del proyecto, como el diseño, el código, la configuración, los datos de prueba y los casos de prueba
  - La mala gestión de defectos y problemas similares pueden dar lugar a defectos acumulados y otras deudas técnicas
- Problemas con el proveedor:
  - Un tercero puede no entregar un producto o servicio necesario, o ir a la quiebra
  - Problemas contractuales pueden causarle problemas al proyecto

Los riesgos del proyecto pueden afectar tanto las actividades de desarrollo como las actividades de prueba. En algunos casos, los jefes de proyecto son responsables de manejar todos los riesgos del proyecto, pero no es inusual que los jefes de prueba tengan la responsabilidad de los riesgos del proyecto relacionados con la prueba.

5.5.3 Pruebas basadas en el riesgo y en la calidad del producto

El riesgo se utiliza para enfocar el esfuerzo requerido durante la prueba. Se utiliza para decidir dónde y cuándo comenzar las pruebas e identificar las áreas que necesitan más atención. Las pruebas se utilizan para reducir la probabilidad de que ocurra un evento adverso, o para reducir el impacto de un evento adverso. Las pruebas se utilizan como una actividad de mitigación de riesgos, para proporcionar información sobre los riesgos identificados, así como para proporcionar información sobre los riesgos residuales (no resueltos).

Un enfoque de prueba basado en el riesgo proporciona oportunidades proactivas para reducir los niveles de riesgo del producto. Implica el análisis del riesgo del producto, que incluye la identificación de los riesgos del producto y la evaluación de la probabilidad y el impacto de cada riesgo. La información sobre el riesgo del producto resultante se utiliza para guiar la planificación de prueba, la especificación, la preparación y la ejecución de los casos de prueba y la monitorización y control de la prueba. El análisis temprano de los riesgos de los productos contribuye al éxito de un proyecto.

En un enfoque basado en el riesgo, los resultados del análisis de riesgo del producto se utilizan para:

- Determinar las técnicas de prueba a emplear
- Determinar los niveles y tipos particulares de pruebas que se realizarán (p. ej., pruebas de seguridad, pruebas de accesibilidad)
- Determinar el alcance de las pruebas a realizar
- Dar prioridad a las pruebas en un intento de encontrar los defectos críticos lo antes posible
- Determinar si se podría emplear alguna actividad además de las pruebas para reducir el riesgo (p. ej., brindar capacitación a diseñadores sin experiencia)

Las pruebas basadas en el riesgo se basan en el conocimiento colectivo y el conocimiento de las partes interesadas del proyecto para llevar a cabo el análisis del riesgo del producto. Para garantizar que se minimice la probabilidad de un fallo del producto, las actividades de administración de riesgos brindan un enfoque por disciplinas para:

- Analizar (y reevaluar de forma regular) lo que puede salir mal (riesgos)
- Determinar qué riesgos son importantes tratar
- Implementar acciones para mitigar esos riesgos
- Hacer planes de contingencia para hacer frente a los riesgos en caso de que se conviertan en eventos reales

Además, las pruebas pueden identificar nuevos riesgos, ayudar a determinar qué riesgos deben mitigarse y disminuir la incertidumbre acerca de los mismos.

5.6 Gestión de Defectos

Dado que uno de los objetivos de las pruebas es encontrar defectos, se deben registrar los defectos encontrados durante las pruebas. La forma en que se registran los defectos puede variar, según el contexto del componente o sistema que se esté probando, el nivel de prueba y el modelo de ciclo de vida de desarrollo de software. Cualquier defecto identificado debe investigarse y debe rastrearse desde el descubrimiento y la clasificación hasta su resolución (p. ej., la corrección de los defectos y la prueba de confirmación exitosa de la solución, el aplazamiento a una entrega posterior, la aceptación como limitación permanente del producto, etc.). Para gestionar todos los defectos a su resolución, una organización debe establecer un proceso de gestión de defectos que incluya un flujo de trabajo y reglas para la clasificación. Este proceso debe acordarse con todos los que participan en la gestión de defectos, incluidos los diseñadores, desarrolladores, probadores y propietarios de productos. En algunas organizaciones, el registro y seguimiento de defectos puede ser muy informal.

Durante el proceso de gestión de defectos, algunos de los informes pueden resultar en la descripción de falsos positivos, no fallos reales debido a defectos. Por ejemplo, una prueba puede fallar cuando una conexión de red se interrumpe o se agota el tiempo de espera. Este comportamiento no se debe a un defecto en el objeto de prueba, pero es una anomalía que debe investigarse. Los probadores deben intentar minimizar el número de falsos positivos reportados como defectos.

Los defectos se pueden informar durante la codificación, el análisis estático, las revisiones, las pruebas dinámicas o el uso de un producto de software. Los defectos pueden informarse para problemas en el código o los sistemas de trabajo, o en cualquier tipo de documentación, incluidos los requisitos, las historias de usuario y los criterios de aceptación, los documentos de desarrollo, los documentos de prueba, los manuales de usuario o las guías de instalación. Para tener un proceso de gestión de defectos eficaz y eficiente, las organizaciones pueden definir normas para los atributos, la clasificación y el flujo de trabajo de los defectos.

Los informes típicos de defectos tienen los siguientes objetivos:

- Proporcionar a los desarrolladores y otras partes información sobre cualquier evento adverso que haya ocurrido, que les permita identificar efectos específicos, aislar el problema con una prueba de reproducción mínima y corregir los posibles defectos, según sea necesario o para resolver el problema
- Proporcionar a los gerentes de pruebas un medio para rastrear la calidad del producto de trabajo y el impacto en las pruebas (p. ej., si se reportan muchos defectos, los probadores habrán pasado mucho tiempo informándolos en lugar de ejecutar pruebas, y habrá necesidad de ejecutar más pruebas de confirmación)
- Proporcionar ideas para el desarrollo y la mejora del proceso de prueba

Un informe de defectos presentado durante las pruebas dinámicas generalmente incluye:

- Un identificador
- Un título y un breve resumen del defecto reportado
- Fecha del informe de defecto, organización emisora y autor
- Identificación del elemento de prueba (elemento de configuración que se está probando) y el entorno
- La (s) fase (s) del ciclo de vida de desarrollo en la(s) que se observó el defecto
- Una descripción del defecto que permita la reproducción y resolución, incluidos registros, capturas de pantalla de volcados de bases de datos o grabaciones (si se encuentran durante la ejecución de la prueba)

- Resultados esperados y reales
- Alcance o grado de impacto (gravedad) del defecto sobre los intereses de las partes interesadas
- Urgencia/prioridad del arreglo
- Estado del informe de defectos (p. ej., abierto, diferido, duplicado, en espera de ser reparado, en espera de pruebas de confirmación, reabierto, cerrado)
- Conclusiones, recomendaciones y aprobaciones
- Problemas globales, como otras áreas que pueden verse afectadas por un cambio como resultado del defecto
- Historial de cambios, como la secuencia de acciones tomadas por los miembros del equipo del proyecto con respecto al defecto para aislarlo, repararlo y confirmarlo como arreglado
- Referencias, incluyendo el caso de prueba que reveló el problema

Algunos de estos detalles pueden incluirse y/o administrarse automáticamente cuando se usan herramientas de gestión de defectos, p. ej., la asignación automática de un identificador, la asignación y actualización del estado del informe de defectos durante el flujo de trabajo, etc. Los defectos encontrados durante las pruebas estáticas, particularmente las revisiones, normalmente se documentan de una manera diferente, p. ej., en las notas de la reunión de revisión.

Se puede encontrar un ejemplo del contenido de un informe de defectos en la norma ISO (ISO/IEC/IEEE 29119-3) (que se refiere a los informes de defectos como informes de incidentes)).

6 Soporte de Herramientas para Pruebas — 40 minutos

Palabras clave

pruebas guiadas por datos, pruebas guiadas por palabra clave, herramienta de pruebas de rendimiento, automatización de la prueba, herramienta de ejecución de pruebas, herramienta de gestión de pruebas

Objetivos de Aprendizaje para las herramientas de prueba

6.1 Consideraciones sobre las herramientas de prueba

FL-6.1.1 (K2) Clasificar las herramientas de prueba según su propósito y las actividades de prueba que apoyan

FL-6.1.2 (K1) Identificar los beneficios y riesgos de la automatización de pruebas

FL-6.1.3 (K1) Recordar las consideraciones especiales para la ejecución de prueba y herramientas de gestión de pruebas

6.2 Uso efectivo de las herramientas

FL-6.2.1 (K1) Identificar los principios principales para seleccionar una herramienta

FL-6.2.2 (K1) Recordar los objetivos para usar proyectos piloto para introducir herramientas

FL-6.2.3 (K1) Identificar los factores de éxito para la evaluación, implementación, despliegue y soporte continuo de las herramientas de prueba en una organización

6.1 Consideraciones sobre las Herramientas de Prueba

Las herramientas de prueba se pueden usar para apoyar una o más actividades de prueba. Tales herramientas incluyen:

- Herramientas que se utilizan directamente en las pruebas, como las herramientas de ejecución de pruebas y las herramientas de preparación de datos de prueba
- Herramientas que ayudan a administrar los requisitos, casos de prueba, procedimientos de prueba, guiones de prueba automatizados, resultados de prueba, datos de prueba y defectos, y para informar y monitorizar la ejecución de la prueba
- Herramientas que se utilizan para la investigación y evaluación
- Cualquier herramienta que ayude en las pruebas (una hoja de cálculo también es una herramienta de prueba en este sentido)

6.1.1 Clasificación de las herramientas de prueba

Las herramientas de prueba pueden tener uno o más de los siguientes propósitos según el contexto:

- Mejorar la eficiencia de las actividades de prueba al automatizar tareas repetitivas o tareas que requieren recursos significativos cuando se realizan manualmente (p. ej., ejecución de pruebas, pruebas de regresión)
- Mejorar la eficiencia de las actividades de prueba al respaldar las actividades de prueba manuales a lo largo del proceso de prueba (consulte la sección 1.4)
- Mejorar la calidad de las actividades de prueba al permitir pruebas más consistentes y un mayor nivel de reproducibilidad de los defectos
- Automatizar actividades que no se pueden ejecutar manualmente (p. ej., pruebas de rendimiento a gran escala)
- Aumentar la confiabilidad de las pruebas (p. ej., automatizando grandes comparaciones de datos o simulando el comportamiento)

Las herramientas se pueden clasificar según varios criterios, como el propósito, la fijación de precios, el modelo de licencia (p. ej., el código comercial o código abierto) y la tecnología utilizada. Las herramientas se clasifican en este programa de acuerdo con las actividades de prueba que apoyan.

Algunas herramientas apoyan claramente solo o principalmente una actividad; otras pueden apoyar más de una actividad, pero se clasifican según la actividad con la que están más estrechamente asociadas. Las herramientas de un solo proveedor, especialmente aquellas que han sido diseñadas para trabajar juntas, pueden proporcionarse como un paquete integrado.

Algunos tipos de herramientas de prueba pueden ser intrusivos, lo que significa que pueden afectar el resultado real de la prueba. Por ejemplo, los tiempos de respuesta reales para una aplicación pueden ser diferentes debido a las instrucciones adicionales que se ejecutan con una herramienta de prueba de rendimiento, o la cantidad de cobertura de código lograda puede estar distorsionada debido al uso de una herramienta de cobertura. La consecuencia de usar herramientas intrusivas se llama efecto sonda.

Algunas herramientas ofrecen soporte que suele ser más adecuado para los desarrolladores (p. ej., herramientas que se utilizan durante las pruebas de componentes e integración). Estas herramientas están marcadas con una “(D)” en las secciones a continuación.

Soporte de herramientas para la gestión de pruebas y productos de prueba

Las herramientas de administración pueden aplicarse a cualquier actividad de prueba durante todo el ciclo de vida del desarrollo del software. Ejemplos de herramientas que admiten la administración de pruebas y

software de prueba incluyen:

- Herramientas de administración de prueba y herramientas de administración del ciclo de vida de la aplicación (ALM)
- Herramientas de gestión de requisitos (p. ej., trazabilidad de los objetos de prueba)
- Herramientas de gestión de defectos
- Herramientas de gestión de configuración
- Herramientas de integración continua (D)

Herramientas de apoyo para pruebas estáticas

Las herramientas de pruebas estáticas están asociadas con las actividades y los beneficios descritos en el capítulo 3. Los ejemplos de estas herramientas incluyen:

- Herramientas de apoyo a las revisiones
- Herramientas de análisis estático (D)

Herramientas de apoyo para el diseño e implementación de pruebas

Las herramientas de diseño de pruebas ayudan en la creación de productos de trabajo mantenibles en el diseño e implementación de pruebas, incluidos casos de prueba, procedimientos de prueba y datos de prueba. Los ejemplos de estas herramientas incluyen:

- Herramientas de diseño de pruebas
- Herramientas de pruebas basadas en modelos
- Herramientas de preparación de datos de prueba
- Herramientas de desarrollo guiado por pruebas de aceptación (ATDD) y desarrollo guiado por el comportamiento (BDD)
- Herramientas de desarrollo guiado por pruebas (TDD) (D)

En algunos casos, las herramientas que admiten el diseño y la implementación de pruebas también pueden respaldar la ejecución y el registro de pruebas, o proporcionar sus resultados directamente a otras herramientas que admiten la ejecución y el registro de pruebas.

Herramientas de apoyo para la ejecución y registro de pruebas

Existen muchas herramientas para apoyar y mejorar las actividades de ejecución y registro de prueba. Ejemplos de estas herramientas incluyen:

- Herramientas de ejecución de prueba (p. ej., para ejecutar pruebas de regresión)
- Herramientas de cobertura (p. ej., cobertura de requisitos, cobertura del código (D))
- Arneses de prueba (D)
- Herramientas de prueba marco unitario (D)

Herramientas de apoyo para medición del desempeño y análisis dinámico

La medición del rendimiento y las herramientas de análisis dinámico son esenciales para respaldar las actividades de pruebas de rendimiento y carga, ya que estas actividades no se pueden realizar de forma manual. Ejemplos de estas herramientas incluyen:

- Herramientas de prueba de rendimiento
- Herramientas de monitorización
- Herramientas de análisis dinámico (D)

Herramientas de apoyo para necesidades de pruebas especializadas

Además de las herramientas que admiten el proceso de prueba general, hay muchas otras herramientas que admiten problemas de prueba más específicos. Ejemplos de estos incluyen herramientas que se centran en:

- Evaluación de la calidad de los datos
- Conversión y migración de datos
- Pruebas de usabilidad
- Pruebas de accesibilidad
- Pruebas de localización
- Pruebas de seguridad
- Pruebas de portabilidad (p. ej., pruebas de software en múltiples plataformas compatibles)

6.1.2 Beneficios y riesgos de la automatización de pruebas

La simple adquisición de una herramienta no garantiza el éxito. Cada nueva herramienta introducida en una organización requerirá un esfuerzo para lograr beneficios reales y duraderos. Existen beneficios y oportunidades potenciales con el uso de herramientas en las pruebas, pero también existen riesgos. Esto es particularmente cierto en el caso de las herramientas de ejecución de pruebas (que a menudo se conoce como automatización de pruebas).

Los beneficios potenciales de usar herramientas para apoyar la ejecución de pruebas incluyen::

- Reducción del trabajo manual repetitivo (p. ej., ejecutar pruebas de regresión, tareas de configuración/eliminación del entorno, volver a ingresar los mismos datos de prueba y verificar los estándares de codificación), lo que ahorra tiempo
- Mayor consistencia y repetibilidad (p. ej., los datos de prueba se crean de manera coherente, las pruebas se ejecutan mediante una herramienta en el mismo orden con la misma frecuencia, y las pruebas se derivan de forma consistente de los requisitos)
- Evaluación más objetiva (p. ej., medidas estáticas, cobertura)
- Acceso más fácil a la información sobre las pruebas (p. ej., estadísticas y gráficos sobre el avance de las pruebas, las tasas de defectos y el rendimiento)

Los riesgos potenciales de usar herramientas para apoyar la ejecución de pruebas incluyen:

- Las expectativas para la herramienta pueden ser poco realistas (incluidas la funcionalidad y la facilidad de uso)
- El tiempo, el costo y el esfuerzo para la introducción inicial de una herramienta pueden ser subestimados (incluyendo la capacitación y experiencia externa)
- El tiempo y el esfuerzo necesarios para lograr beneficios significativos y continuos de la herramienta pueden ser subestimados (incluida la necesidad de cambios en el proceso de prueba y la mejora continua en la forma en que se utiliza la herramienta)
- El esfuerzo requerido para mantener los recursos de prueba generados por la herramienta puede ser subestimado
- Se puede confiar demasiado en la herramienta (vista como un reemplazo para el diseño o la ejecución de la prueba, o el uso de pruebas automatizadas donde las pruebas manuales serían mejores)
- El control de versión de los recursos de prueba puede ser desatendido

- Los problemas de relaciones e interoperabilidad entre las herramientas críticas pueden ser ignorados, como son las herramientas de gestión de requisitos, herramientas de gestión de configuración, herramientas de gestión de defectos y herramientas de múltiples proveedores
- El proveedor de la herramienta puede cerrar su negocio, retirar la herramienta o vender la herramienta a un proveedor diferente
- El proveedor puede proporcionar una respuesta deficiente para soporte, actualizaciones y correcciones de defectos
- Un proyecto de código abierto puede ser suspendido
- Es posible que la herramienta no admita una nueva plataforma o tecnología
- Es posible que no haya una propiedad clara de la herramienta (p. ej., para tutorías, actualizaciones, etc.)

6.1.3 Consideraciones especiales para la ejecución de prueba y herramientas de gestión de pruebas

Para tener una implementación sin problemas y exitosa, hay una serie de cosas que deben tenerse en cuenta al seleccionar e integrar las herramientas de ejecución de pruebas y de gestión de pruebas en una organización.

Herramientas de ejecución de pruebas

Las herramientas de ejecución de pruebas ejecutan objetos de prueba utilizando guiones de prueba automatizados. Este tipo de herramienta a menudo requiere un esfuerzo significativo para lograr beneficios significativos.

La captura de pruebas mediante el registro de las acciones de un probador manual parece atractiva, pero este enfoque no se ajusta a un gran número de guiones de prueba. Un guión capturado es una representación lineal con datos y acciones específicos como parte de cada guión. Este tipo de guión puede ser inestable cuando ocurren eventos inesperados. La última generación de estas herramientas, que aprovecha la tecnología de captura de imágenes “inteligente”, ha aumentado la utilidad de esta clase de herramientas, aunque los guiones generados aún requieren un mantenimiento continuo a medida que la interfaz de usuario del sistema evoluciona con el tiempo.

Un enfoque de pruebas guiadas por datos separa las entradas de prueba y los resultados esperados, generalmente en una hoja de cálculo, y utiliza un guión de prueba más genérico que puede leer los datos de entrada y ejecutar el mismo guión de prueba con datos diferentes. Los probadores que no están familiarizados con el lenguaje de guión pueden crear nuevos datos de prueba para estos guiones predefinidos.

En un enfoque de prueba guiado por palabras clave, un guión genérico procesa palabras clave que describen las acciones que se deben tomar (también llamadas palabras de acción), que luego llaman guiones de palabras clave para procesar los datos de prueba asociados. Los probadores (incluso si no están familiarizados con el lenguaje de guión) pueden definir pruebas utilizando las palabras clave y los datos asociados, que se pueden adaptar a la aplicación que se está probando. En ISTQB-TAE Programa de Estudio – Nivel Avanzado – Automatización de la Prueba, Fewster 1999 y Buwalda 2001 se proporcionan más detalles y ejemplos de métodos de prueba guiados por datos y por palabras clave.

Los enfoques anteriores requieren que alguien tenga experiencia en el lenguaje de guión (probadores, desarrolladores o especialistas en automatización de pruebas). Independientemente de la técnica de escritura de guiones utilizada, los resultados esperados para cada prueba deben compararse con los resultados reales de la prueba, ya sea dinámicamente (mientras la prueba se está ejecutando) o almacenados para una comparación posterior (posterior a la ejecución).

Las herramientas de prueba basadas en modelos (PBM) permiten capturar una especificación funcional en forma de modelo, como un diagrama de actividad. Esta tarea generalmente la realiza un diseñador de

sistemas. La herramienta PBM interpreta el modelo para crear especificaciones de casos de prueba que luego pueden guardarse en una herramienta de administración de pruebas y/o ejecutarse mediante una herramienta de ejecución de prueba (consulte ISTQB-MBT Programa de Estudio – Nivel Básico – Pruebas Basadas en Modelos).

Herramientas de gestión de pruebas

Las herramientas de gestión de pruebas a menudo necesitan interactuar con otras herramientas u hojas de cálculo por varias razones, entre ellas:

- Producir información útil en un formato que se ajuste a las necesidades de la organización
- Mantener una trazabilidad constante a los requisitos en una herramienta de gestión de requisitos.
- Vincularla con la información de la versión del objeto de prueba en la herramienta de gestión de configuración

Esto es particularmente importante tenerlo en cuenta cuando se usa una herramienta integrada (p. ej., Gestión del Ciclo de Vida de la Aplicación), que incluye un módulo de gestión de pruebas (y posiblemente un sistema de gestión de defectos), así como otros módulos (p. ej., información sobre el calendario y presupuesto del proyecto) utilizado por diferentes grupos dentro de una organización.

6.2 Uso Efectivo de las Herramientas

6.2.1 Principios principales para la selección de herramientas

Las principales consideraciones al seleccionar una herramienta para una organización incluyen:

- Evaluar la madurez de una organización, sus fortalezas y debilidades
- Identificar las oportunidades para un proceso de prueba mejorado apoyado por herramientas
- Comprender las tecnologías utilizadas por el (los) objeto (s) de prueba para seleccionar una herramienta que sea compatible con esa tecnología
- Las herramientas de compilación e integración continua ya en uso dentro de la organización, para garantizar la compatibilidad e integración de la herramienta
- Evaluar la herramienta contra requisitos claros y criterios objetivos
- Tener en cuenta si la herramienta está disponible o no para un período de prueba gratuito (y por cuánto tiempo)
- Evaluación del proveedor (incluidos aspectos de capacitación, soporte y comerciales) o soporte para herramientas no comerciales (p. ej., de código abierto)
- Identificar los requisitos internos para el adiestramiento y tutoría en el uso de la herramienta
- Evaluar las necesidades de capacitación, teniendo en cuenta las habilidades de prueba (y automatización de prueba) de quienes trabajarán directamente con la (s) herramienta (s)
- Considerar las ventajas y desventajas de varios modelos de licencia (p. ej., comercial o de código abierto)
- Estimación de una relación costo-beneficio basada en un estudio de viabilidad comercial concreto (si es necesario)

Como paso final, se debe realizar una evaluación de prueba de concepto para establecer si la herramienta funciona de manera efectiva con el software a prueba y dentro de la infraestructura actual o, si es necesario, identificar los cambios necesarios en esa infraestructura para usar la herramienta de manera efectiva.

6.2.2 Proyectos piloto para introducir una herramienta en una organización

Después de completar la selección de la herramienta y una prueba de concepto exitosa, la introducción de la herramienta seleccionada en una organización generalmente comienza con un proyecto piloto, que tiene los siguientes objetivos:

- Obtener un conocimiento profundo de la herramienta, comprendiendo tanto sus puntos fuertes como sus puntos débiles
- Evaluar cómo la herramienta se ajusta a los procesos y prácticas existentes, y determinar qué debería cambiar
- Decidir formas estándar de utilizar, administrar, almacenar y mantener la herramienta y los activos de prueba (p. ej., decidir sobre convenciones de nombres para archivos y pruebas, crear bibliotecas y definir la modularidad de juegos de prueba)
- Evaluar si los beneficios se obtendrán a un costo razonable
- Comprender las métricas que desea que la herramienta recopile e informar, y configurar la herramienta para garantizar que estas métricas se puedan capturar e informar

6.2.3 Factores de éxito para las herramientas

Los factores de éxito para la evaluación, implementación, despliegue y soporte continuo de las herramientas en una organización incluyen:

- Introducir la herramienta gradualmente para el resto de la organización
- Adaptar y mejorar los procesos para adecuarlos al uso de la herramienta
- Proporcionar capacitación, entrenamiento y tutoría para usuarios de herramientas
- Definir pautas para el uso de la herramienta (p. ej., normas internas para la automatización)
- Implementar una forma de recopilar información de uso del uso real de la herramienta
- Uso y beneficios de la herramienta de monitorización
- Brindar apoyo a los usuarios de una herramienta dada
- Recopilar las lecciones aprendidas de todos los usuarios

También es importante asegurarse de que la herramienta esté integrada técnica y organizativamente en el ciclo de vida de desarrollo del software, lo que puede involucrar a organizaciones independientes responsables de las operaciones y/o proveedores externos.

Consulte Graham 2012 para conocer las experiencias y consejos sobre el uso de herramientas de ejecución de pruebas.

7 Referencias

Normas

ISO/IEC/IEEE 29119-1 (2013) Software and systems engineering - Software testing - Part 1: Concepts and definitions

ISO/IEC/IEEE 29119-2 (2013) Software and systems engineering - Software testing - Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2013) Software and systems engineering - Software testing - Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2015) Software and systems engineering - Software testing - Part 4: Test techniques

ISO/IEC 25010, (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models

ISO/IEC 20246: (2017) Software and systems engineering — Work product reviews

UML 2.5, Unified Modeling Language Reference Manual, http://www.omg.org/spec/UML/2.5.1/, 2017

Documentos de la ISTQB

Glosario de la ISTQB

ISTQB Descripción General del Nivel Básico 2018

ISTQB-MBT Programa de Estudio – Nivel Básico – Extensión – Pruebas Basadas en Modelos

ISTQB-AT Programa de Estudio – Nivel Básico – Estensión – Probador Ágil

ISTQB-ATA Programa de Estudio – Nivel Avanzado – Analista de Pruebas

ISTQB-ATM Programa de Estudio – Nivel Avanzado – Jefe de Pruebas

ISTQB-SEC Programa de Estudio – Nivel Avanzado – Pruebas de Seguridad

ISTQB-TAE Programa de Estudio – Nivel Avanzado – Ingeniero de Automatización de Pruebas

ISTQB-ETM Programa de Estudio – Nivel Experto – Gestión de Pruebas

ISTQB-EITP Programa de Estudio – Nivel Experto – Proceso de Prueba

Libros y Artículos

Beizer, B. (1990*) Software Testing Techniques (2e)*, Van Nostrand Reinhold: Boston MA

Black, R. (2017) *Agile Testing Foundations*, BCS Learning & Development Ltd: Swindon UK

Black, R. (2009) *Managing the Testing Process (3e)*, John Wiley & Sons: New York NY

Buwalda, H. et al. (2001) *Integrated Test Design and Automation*, Addison Wesley: Reading MA

Copeland, L. (2004) *A Practitioner's Guide to Software Test Design*, Artech House: Norwood MA

Craig, R. and Jaskiel, S. (2002) *Systematic Software Testing*, Artech House: Norwood MA

Crispin, L. and Gregory, J. (2008) *Agile Testing*, Pearson Education: Boston MA

Fewster, M. and Graham, D. (1999) *Software Test Automation*, Addison Wesley: Harlow UK

Gilb, T. and Graham, D. (1993) *Software Inspection*, Addison Wesley: Reading MA

Graham, D. and Fewster, M. (2012) *Experiences of Test Automation*, Pearson Education: Boston MA

Gregory, J. and Crispin, L. (2015) *More Agile Testing*, Pearson Education: Boston MA

Jorgensen, P. (2014) *Software Testing, A Craftsman's Approach (4e)*, CRC Press: Boca Raton FL

Kaner, C., Bach, J. and Pettichord, B. (2002) *Lessons Learned in Software Testing*, John Wiley & Sons: New York NY

Kaner, C., Padmanabhan, S. and Hoffman, D. (2013) *The Domain Testing Workbook*, Context-Driven Press: New York NY

Kramer, A., Legeard, B. (2016) *Model-Based Testing Essentials: Guide to the ISTQB Certified Model-Based Tester: Foundation Level*, John Wiley & Sons: New York NY

Myers, G. (2011) *The Art of Software Testing*, (3e), John Wiley & Sons: New York NY

Sauer, C. (2000) "The Effectiveness of Software Development Technical Reviews: A Behaviorally Motivated Program of Research," *IEEE Transactions on Software Engineering*, *Volume 26, Issue 1, pp 1-*

Shull, F., Rus, I., Basili, V. July 2000. "How Perspective-Based Reading can Improve Requirement Inspections." *IEEE Computer, Volume 33, Issue 7, pp 73-79*

van Veenendaal, E. (ed.) (2004) *The Testing Practitioner* (Chapters 8 - 10), UTN Publishers: The Netherlands

Wiegers, K. (2002) *Peer Reviews in Software*, Pearson Education: Boston MA

Weinberg, G. (2008) *Perfect Software and Other Illusions about Testing*, Dorset House: New York NY

Otros recursos (no mencionados directamente en este Programa de Estudio)

Black, R., van Veenendaal, E. and Graham, D. (2012) *Foundations of Software Testing: ISTQB Certification (3e)*, Cengage Learning: London UK

Hetzel, W. (1993) *Complete Guide to Software Testing (2e)*, QED Information Sciences: Wellesley MA

Spillner, A., Linz, T., and Schaefer, H. (2014) *Software Testing Foundations (4e)*, Rocky Nook: San Rafael CA

8 Apéndice A – Información Básica del Programa de Estudio

Historia de este Documento

Este documento es el ISTQB Programa de estudio de Nivel Básico de Probadores Certificados, la cualificación internacional de primer nivel aprobada por la ISTQB (www.istqb.org).

Este documento se preparó entre 2014 y 2018 por un grupo de trabajo compuesto por miembros designados por la Junta Internacional de Calificación de Pruebas de Software (ISTQB). La versión de 2018 fue revisada inicialmente por representantes de todas las juntas de miembros de la ISTQB, y luego por representantes de la comunidad internacional de pruebas de software.

Objetivos de la Calificación para el Certificado de Nivel Básico

- Obtener reconocimiento por las pruebas como una especialización de ingeniería de software esencial y profesional
- Proporcionar un marco estándar para el desarrollo de las carreras de los probadores
- Permitir que los probadores calificados profesionalmente sean reconocidos por empleadores, clientes y compañeros, y elevar el perfil de los mismos
- Promover prácticas de prueba consistentes y buenas dentro de todas las disciplinas de la ingeniería de software
- Identificar temas de prueba que sean relevantes y de valor para la industria
- Permitir que los proveedores de software contraten probadores certificados y, por lo tanto, obtengan una ventaja comercial sobre sus competidores publicitando su política de contratación de probadores
- Brindar una oportunidad a los probadores y aquellos interesados en las pruebas de adquirir una calificación reconocida internacionalmente en la materia

Objetivos de la Calificación Internacional

- Poder comparar el conocimiento de las pruebas en diferentes países
- Permitir que los probadores se muevan a través de las fronteras de los países más fácilmente
- Permitir que los proyectos multinacionales/internacionales tengan un entendimiento común de los problemas relacionados con las pruebas
- Aumentar el número de probadores calificados en todo el mundo
- Tener mayor impacto/valor como una iniciativa internacional que desde cualquier enfoque específico de país
- Desarrollar un organismo internacional común de comprensión y conocimiento sobre las pruebas a través del programa y la terminología, y aumentar el nivel de conocimiento sobre las pruebas para todos los participantes
- Promover las pruebas como profesión en más países
- Permitir que los probadores obtengan una calificación reconocida en su idioma nativo
- Permitir el intercambio de conocimientos y recursos entre países
- Proporcionar reconocimiento internacional a los probadores y a esta cualificación debido a la participación de muchos países

Condiciones de Admisión para esta Cualificación

El criterio de ingreso para tomar el examen de la ISTQB Probador Certificado Nivel Básico, es que los candidatos tengan interés en las pruebas de software. Sin embargo, se recomienda encarecidamente que los candidatos también:

- Tengan al menos un mínimo de experiencia en desarrollo de software o pruebas de software, como son seis meses de experiencia como comprobador de aceptación de sistemas o usuarios o como desarrollador de software
- Tomen un curso que haya sido acreditado por una de las juntas de miembros reconocidas por la ISTQB, según las normas de la ISTQB.

Información e Historia de la Certificación 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 (p. ej., del ISEB, ASQF o un comité nacional reconocido de la 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 de la ISTQB. La 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 capacitación y la realización de exámenes.

9 Apéndice B - Objetivos de Aprendizaje/Nivel Cognitivo del Conocimiento

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 asignad.

Nivel 1: Recordar (K1)

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

Palabras clave: Identificar, recordar, recuperar, retener, reconocer, conocer

Ejemplos:

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: Comprender (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 la razón por la cual las pruebas deben diseñarse lo antes posible:

- Localizar defectos cuando 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: los objetos de prueba tanto para la prueba de integración como para las pruebas de sistema incluyen más de un componente, y tanto la prueba de integración como las pruebas de sistemas pueden incluir tipos de prueba no funcionales
- Diferencias: las pruebas de integración se concentran en las interfaces e interacciones, y las pruebas de sistema se concentran en aspectos de todo el sistema, como el procesamiento de extremo a extremo

Nivel 3: Aplicar (K3)

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

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

Ejemplos:

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

Referencia (Para los niveles cognitivos de los objetivos de aprendizaje)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon: Boston MA

10 Apéndice C - Notas de la Versión

ISTQB Programa de Estudio Básico 2018 es una actualización y re-escritura importante de la versión de 2011. Por este motivo, no hay notas detalladas de la versión por capítulo y sección. Sin embargo, aquí se proporciona un resumen de los cambios principales. Además, en un documento separado de Notas de la versión, la ISTQB proporciona trazabilidad entre los objetivos de aprendizaje en la versión 2011 del Programa de estudios de nivel básico y los objetivos de aprendizaje en la versión de 2018 del Programa de estudios de nivel básico, que muestran cuáles se han agregado, actualizado o eliminado.

A comienzos de 2017, más de 550,000 personas en más de 100 países han tomado el examen básico y más de 500,000 son probadores certificados en todo el mundo. ¡Con la expectativa de que todos ellos hayan leído el Programa de Estudio Básico para poder aprobar el examen, esto hace que el Programa de Estudio Básico sea el documento de prueba de software más leído en toda la vida!

Esta importante actualización se realiza con respecto a este patrimonio y para mejorar el valor que la ISTQB brinda a las siguientes 500,000 personas en la comunidad global de pruebas.

En esta versión, todos los objetivos de aprendizaje se han editado para hacerlos atómicos, y para crear una trazabilidad clara desde cada objetivo de aprendizaje a las secciones de contenido (y preguntas del examen) que están relacionadas con ese objetivo de aprendizaje, y para tener una trazabilidad clara desde las secciones de contenido (y las preguntas del examen) de vuelta al objetivo de aprendizaje asociado. Además, las asignaciones de tiempo al capítulo se han hecho más realistas que las de la versión 2011 del programa de estudio, mediante el uso de heurísticas y fórmulas probadas utilizadas con otros programas de la ISTQB, que se basan en un análisis de los objetivos de aprendizaje que se cubrirán en cada capítulo.

Si bien este es un Programa de Estudio Básico, que expresa las mejores prácticas y técnicas que han resistido la prueba del tiempo, hemos realizado cambios para modernizar la presentación del material, especialmente en términos de métodos de desarrollo de software (p. ej., Scrum y despliegue continuo) y tecnologías (p. ej., el internet de las cosas). Hemos actualizado las normas de referencia para hacerlas más recientes de la siguiente manera:

1. La ISO/IEC/IEEE 29119 reemplaza a la norma IEEE 829.
2. La ISO/IEC 25010 reemplaza a la norma ISO 9126.
3. La ISO/IEC 20246 reemplaza a la norma IEEE 1028.

Además, dado que la cartera de clientes de la ISTQB ha crecido dramáticamente en la última década, hemos agregado referencias cruzadas extensas al material relacionado en otros programas de estudio de la ISTQB, cuando es relevante, y también hemos revisado cuidadosamente la alineación con todos los programas de estudio y con el Glosario de la ISTQB. El objetivo es hacer que esta versión sea más fácil de leer, comprender, aprender y traducir, centrándose en aumentar la utilidad práctica y el equilibrio entre el conocimiento y las habilidades

Para obtener un análisis detallado de los cambios realizados en esta versión, consulte ISTQB Descripción General del Nivel Básico de Probador Certificado de 2018.

11 Índice