Probador Certificado del ISTQB®

Programa de Estudio de Nivel Básico

Traducción realizada por el
Spanish Software Testing Qualifications Board con el apoyo del
Hispanic America Software Testing Qualifications Board
Versión ES 001.32
Basada en el Programa de Estudio
“Certified Tester - Foundation Level, Version 2018”

International Software Testing Qualifications Board

Nota sobre Derechos de Propiedad Intelectual.

Historial de Revisiones

Versión	Fecha	Observaciones
ISTQB 2018	27-Abril-2018	Versión candidata a la publicación.
ISTQB 2018	12-Febrero-2018	Versión beta candidata.
ISTQB 2018	19-Enero-2018	Revisión cruzada, versión interna 3.0.
ISTQB 2018	15-Enero-2018	Pre-revisión cruzada de la versión interna 2.9, incorpora modificaciones del Equipo Principal ("Core Team").
ISTQB 2018	9-Diciembre-2017	Publicación de la revisión Alpha 2.5 - Edición técnica de la publicación 2.0, no se han añadido nuevos contenidos.
ISTQB 2018	22-Noviembre-2017	Publicación de la revisión Alpha 2.0 - Actualización Mayor del Programa de Estudio - Probador Certificado - Nivel Básico 2018 - ver Apéndice C - Para más detalles, consultar las Notas de la Publicación ("Release Notes").
ISTQB 2018	12-Junio-2017	Publicación de la revisión Alpha - Actualización Mayor del Programa de Estudio - Probador Certificado - Nivel Básico 2018 - ver Apéndice C - Para más detalles, consultar las Notas de la Publicación ("Release Notes").
ISTQB 2011	1-Abril-2011	Publicación de Mantenimiento del Programa de Estudio - Probador Certificado - Nivel Básico - Para más detalles, consultar las Notas de la Publicación ("Release Notes").
ISTQB 2010	30-Marzo-2010	Publicación de Mantenimiento del Programa de Estudio - Probador Certificado - Nivel Básico - Para más detalles, consultar las Notas de la Publicación ("Release Notes").
ISTQB 2007	01-Mayo-2007	Publicación de Mantenimiento del Programa de Estudio - Probador Certificado - Nivel Básico.
ISTQB 2005	01-Julio-2005	Programa de Estudio - Probador Certificado - Nivel Básico.
ASQF V2.2	Julio-2003	ASQF Programa de Estudio - Nivel Básico 2.2, “Lehrplan Grundlagen des Software-testens“.
ISEB V2.0	25-Febrero-1999	ISEB Programa de Estudio de Fundamentos de Pruebas de Software V2.0.

Tabla de Contenidos

Agradecimientos

Este documento fue hecho público formalmente por la Asamblea General del ISTQB, el 4 de junio de 2018.

Fue desarrollado por un equipo del International Software Testing Qualifications Board: Klaus Olsen (presidente), Tauhida Parveen (vicepresidenta), Rex Black (jefe 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 agradece a Rex Black y Dorothy Graham por su edición técnica, y al equipo de revisión, al equipo de revisión cruzada y a los Comités Miembro por sus sugerencias y aportaciones.

Las siguientes personas participaron en la revisión, en los comentarios y en la 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, Gábor Kapros, Beata Karpinska, Karl Kemminger, Kwanho Kim, Seonjoon 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 Majernik, Rik Marselis, Romanos Matthaios, Judy McKay, Fergus McLachlan, Dénes Medzihradszky, Stefan Merkel, Armin Metzger, Don Mills, Gary Mogyorodi, Ninna Morin, Ingvar Nordström, Adam Novak, Avi Ofer, Magnus C Ohlsson, Joel Oliviera, Monika Stocklein Olsen, Kenji Onishi, 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, Miroslav 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 Weymouth, Hyungjin Yoon, John Young, Surong Yuan, Ester Zabar, y Karolina Zmitrowicz.

Grupo de Trabajo del Programa de Estudio de Nivel Básico (Edición 2018) del International Software Testing Qualifications Board: Klaus Olsen (chair), Tauhida Parveen (vice chair), Rex Black (project manager), 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, and Stevan Zivanovic. El equipo principal agradece al equipo de revisión y a todos los Comités Miembro por sus sugerencias.

Grupo de trabajo del International Software Testing Qualifications Board Nivel Básico (Edición 2011): Thomas Müller (presidente), Debra Friedenberg. El equipo principal agradece al equipo de revisión (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkkönen, Eric Riou du Cosquier Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal), y a todos los Comités Miembro por sus sugerencias.

Grupo de trabajo del International Software Testing Qualifications Board Nivel Básico (Edición 2010): Thomas Müller (presidente), Rahul Verma, Martin Klonk y Armin Beer. El equipo principal 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 todos los Comités Miembro por sus sugerencias.

Grupo de trabajo del International Software Testing Qualifications Board Nivel Básico (Edición 2007): Thomas Müller (presidente), Dorothy Graham, Debra Friedenberg y Erik van Veenendaal. El equipo principal agradece al equipo de revisión (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson y Wonil Kwon) y a todos los Comités Miembro por sus sugerencias.

Grupo de trabajo del International Software Testing Qualifications Board Nivel Básico (Edición 2005): Thomas Müller (presidente), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson y Erik van Veenendaal. El equipo principal agradece al equipo de revisión y a todos los Comités Miembro por sus sugerencias.

Notas de la Versión en Idioma Español

El Spanish Software Testing Qualifications Board (SSTQB) ha llevado a cabo la traducción del Programa de Estudio de “ISTQB® Certified Tester, Foundation Level” versión de 2018. Este Programa de Estudio se denomina, en idioma español, “Probador Certificado del ISTQB®” de “Nivel Básico” versión 2018.

El equipo de traducción y revisión para este programa de estudio es el siguiente (por orden alfabético):

Revisor:	Aurelio Gandarillas (España)
Responsable de la traducción:	Gustavo Márquez Sosa (España)
Revisor:	José Antonio Rodríguez Gómez (España)
Revisora:	Luisa Morales Gómez Tejedor (España)

El Comité Ejecutivo del SSTQB agradece especialmente las aportaciones de los revisores.

En una siguiente versión se podrán incorporar aportaciones adicionales. El SSTQB considera conveniente mantener abierta la posibilidad de realizar cambios en el “Programa de Estudio”.

Madrid, 24 de febrero de 2019

Traducciones Específicas, Definiciones, Acrónimos, Abreviaturas y Notas

En este capítulo se presentarán traducciones específicas, definiciones, acrónimos, abreviaturas y notas de términos y conceptos que no forman parte del Glosario de Términos del ISTQB y tampoco están incluidos en su traducción al idioma español o castellano. Se ha incorporado este apartado para facilitar la lectura y comprensión de esta “Programa de Estudio”.

Las “Traducciones Específicas, Definiciones, Acrónimos, Abreviaturas y Notas” de este apartado están identificados con el texto **Consultar **** en el pie de página. Sólo se identificará la primera ocurrencia del elemento.

Es conveniente observar que las traducciones de los distintos términos, sean o no del glosario, se han realizado con el objetivo de evitar conflictos en sus traducciones y siguiendo las normas de traducción del SSTQB. Algunos conflictos surgen en el momento de traducir un nuevo término o un nuevo programa de estudio.

Traducciones específicas		
Español	**Inglés**	**Observaciones**
acción de orientación	guiding action	No hay observaciones.
apariencia corporativa	company-wide look-and-feel	No hay observaciones.
ascendente	bottom-up	No hay observaciones.
aseguramiento de la calidad	quality assurance	No hay observaciones.
asesoramiento	mentoring	Este término también se puede traducir como "orientación" u “orientación profesional”.
canal de entrega	delivery pipeline	No hay observaciones.
cartera del producto	product backlog	Este término también se puede traducir como "pila del producto" o "lista del producto".
competencia	skill	Este término también se puede traducir como "capacidad".
comprobación entre amigos	buddy check	No hay observaciones.
condición de guarda	guard condition	No hay observaciones.
corregir	fix	Este término también se puede traducir como "corrección”.
cuestión	issue	Este término también se puede traducir como "asunto" o "tema". En este programa de estudio no se traducirá como "problema". Según el diccionario de la RAE el término "cuestión" significa: 1. f. Pregunta que se hace con intención dialéctica para averiguar la verdad de algo. 3. f. Punto o materia dudoso o discutible. 4. f. Asunto o materia.

Traducciones específicas		
Español	**Inglés**	**Observaciones**
		5. f. Problema que debe ser resuelto por métodos científicos. 6. f. Der. cuestión de tormento. 7. f. p. us. Oposición de términos lógicos o de razones respecto a un mismo tema, que exigen detenido estudio para resolver con acierto.
daño	harm	No hay observaciones.
de forma inequívoca	unambiguously	No hay observaciones.
decisión informada	informed decision	Es una decisión basada en hechos o información.
definición de hecho	definition of done	No hay observaciones.
definición de preparado	definition of ready	No hay observaciones.
Desarrollo Guiado por el Comportamiento (BGC)	Behavior Driven Development (BDD)	No hay observaciones.
Desarrollo Guiado por Pruebas (DGP)	Test Driven Development (TDD)	No hay observaciones.
Desarrollo Guiado por Prueba de Aceptación (DGPA)	Acceptance Test Driven Development (ATDD)	No hay observaciones.
descendente	top-down	No hay observaciones.
desfase	gap	Este término también se puede traducir como "vacío", “brecha”, “diferencia” o “deficiencia”. Este término puede distintas traducciones que dependen del contexto.
desplazamiento hacia la izquierda	shift left	No hay observaciones.
deuda técnica	technical debt	No hay observaciones.
dirección	management	No hay observaciones.
educción	elicitation	Fuente: Glosario de términos del International Requirements Engineering Board (IREB).
efecto sonda	probe effect	No hay observaciones.
emparejamiento	pairing	No hay observaciones.
enfoque disciplinado	disciplined approach	No hay observaciones.
ensayo	dry run	No hay observaciones.
entrenamiento	coaching	Este término también se puede traducir como "entrenamiento profesional".
escriba	scribe	No hay observaciones.
establecer un calendario	schedule	No hay observaciones.
externalización	outsourcing	No hay observaciones.
falso	false	No hay observaciones.
gestión del ciclo de vida de la aplicación	application lifecycle management	No hay observaciones.

Traducciones específicas		
Español	**Inglés**	**Observaciones**
Gestión del Ciclo de Vida de la Aplicación	Application Lifecycle Management	No hay observaciones.
guion capturado	captured script	No hay observaciones.
hoja de sesión de prueba	test session sheet	No hay observaciones.
indicador clave de desempeño	key performance indicator	No hay observaciones.
internalización	insourcing	No hay observaciones.
Internet de las Cosas	Internet of Things	No hay observaciones.
liberar	release	Este término también se puede traducir como "publicar", “publicación”, “lanzar” o lanzamiento.
no	no	No hay observaciones.
obtener	derive	No hay observaciones.
operación	operation	Según el diccionario de la RAE el término "operación" significa: 1. f. Acción y efecto de operar. 2. f. Ejecución de algo.
plazo	timeframe	No hay observaciones.
practicar	exercise	Este término también se puede traducir como "ejercer", "aplicar" o "hacer efectivo". En el contexto actual "practicar la condición de prueba" significa que la ejecución del objeto de prueba recorre el fragmento de código que permite alcanzar la condición de prueba.
prestación	feature	No hay observaciones.
propiedad	ownership	Este término también se puede traducir como "titularidad".
propietario de producto	product owner	Este término también se puede traducir como "dueño de producto".
proyecto de código abierto	open source project	No hay observaciones.
reconstrucción	rework	Este término también se puede traducir como " reelaborar" o “reelaboración”.
recopilar	collate	No hay observaciones.
rentabilidad	cost-effectiveness	No hay observaciones.
retención de datos	data-retention	"retención de datos" se refiere, por lo general, al almacenamiento de datos. La retención de datos podría estar regulada por una legislación, normativa y/o política específicas.
retroalimentación	feedback	No hay observaciones.
reunión de pie	stand-up meeting	No hay observaciones.
revaluar	re-evaluate	Según el diccionario de la RAE: 1. tr. Volver a evaluar.

Traducciones específicas		
Español	**Inglés**	**Observaciones**
ser objeto de seguimiento	be tracked	No hay observaciones.
sí	yes	No hay observaciones.
solución en caliente	hot fix	No hay observaciones.
sopesar	weighing	Según el diccionario de la RAE: 1. tr. Levantar algo como para tantear el peso que tiene o para reconocerlo. 3. tr. Examinar con atención el pro y el contra de un asunto.
susceptibilidad a desbordamientos de la memoria intermedia	susceptibility to buffer overflows	No hay observaciones.
tabla de decisión completa	full decision table	No hay observaciones.
tablero de tareas	task board	Este término también se puede traducir como "panel de tareas".
técnica basada en expertos	expert-based technique	No hay observaciones.
técnica basada en métricas	metrics-based technique	No hay observaciones.
verdadero	true	No hay observaciones.

Tabla 1 - Traducciones Específicas

Definiciones, Acrónimos y Abreviaturas			
Tipo de Término	**Idioma del Acrónimo**	**Término**	**Descripción**
ABREVIATURA	Español	N/A	**N**o **A**plica
ABREVIATURA	Inglés	UX	**U**ser e**X**perience
ACRÓNIMO	Inglés	ALM	**A**pplication **L**ifecycle **M**anagement
ACRÓNIMO	Inglés	ATDD	**A**cceptance **T**est **D**riven **D**evelopment
ACRÓNIMO	Inglés	BDD	**B**ehavior **D**riven **D**evelopment
ACRÓNIMO	Inglés	DoD	**D**efinition **o**f **D**one
ACRÓNIMO	Español	F	**F**also
ACRÓNIMO	Inglés	IoT	**I**nternet **o**f **T**hings
ACRÓNIMO	Inglés	KPI	**k**ey **p**erformance **i**ndicator
ACRÓNIMO	Español	N	**N**o
ACRÓNIMO	Inglés	QA	**Q**uality **A**ssurance
ACRÓNIMO	Español	S	**S**í
ACRÓNIMO	Inglés	TDD	**T**est **D**riven **D**evelopment
ACRÓNIMO	Español	V	**V**erdadero
DEFINICIÓN	Español	ad hoc	Según el diccionario de la RAE:

Definiciones, Acrónimos y Abreviaturas			
Tipo de Término	**Idioma del Acrónimo**	**Término**	**Descripción**
			1. expr. U. para referirse a lo que se dice o hace solo para un fin determinado. 2. loc. adj. Adecuado, apropiado, dispuesto especialmente para un fin.
DEFINICIÓN	Español	compleción	Según el diccionario de la RAE: 1. f. p. us. Acción y efecto de completar. 2. f. p. us. Cualidad de completo.

Tabla 2 – Definiciones, Acrónimos y Abreviaturas

Notas
"persona" es un modelo descriptivo de los usuarios que se utiliza en "Diseño de la Interacción".

Tabla 3 – Notas

0 Introducción

0.1 Objetivo de este Programa de Estudio

Este programa de estudio constituye la base para la formación como Probador Certificado del ISTQB® de Nivel Básico. El ISTQB proporciona este programa de estudio en los siguientes términos:

1. A los comités miembro, para traducir a su idioma local y para acreditar a los proveedores de formación. Los comités miembro pueden adaptar el programa de estudio a sus necesidades lingüísticas particulares y añadir referencias para adaptarlo a sus publicaciones locales.
2. A los organismos de certificación, para elaborar las preguntas del examen en su lengua local adaptadas a los objetivos de aprendizaje de este programa de estudio.
3. A los proveedores de formación, para desarrollar material didáctico y determinar los métodos de enseñanza adecuados.
4. A los candidatos a la certificación, para que preparen el examen de certificación (ya sea como parte de un curso de formación o de forma independiente).
5. A la comunidad internacional de ingeniería de software y sistemas, para avanzar en la profesión de prueba del software y sistemas, y como fuente de libros y artículos.

El ISTQB puede permitir que otras entidades utilicen este programa de estudio para otros fines, siempre y cuando soliciten y obtengan permiso previo y por escrito por parte del ISTQB.

0.2 El Nivel Básico de Probador Certificado en la Prueba de Software

La certificación de Nivel Básico está dirigida a cualquier persona que se encuentre involucrada en la prueba de software. Incluidas las personas que asumen los roles de probadores, analistas de pruebas, ingenieros de pruebas, consultores de pruebas, jefes de prueba, probadores que participan en la prueba de aceptación y desarrolladores de software. Esta cualificación de nivel básico también es apropiada para cualquier persona que desee una comprensión básica de la prueba de software, como propietarios de producto[1], jefes de proyecto, responsables en el área de calidad, jefes de desarrollo de software, analistas de negocio, directores de TI y consultores de gestión. Los titulares del Certificado Básico podrán acceder a las certificaciones de nivel superior en la prueba de software.

El documento **ISTQB Foundation Level Overview 2018** es un documento adicional que incluye la siguiente información:

- Resultados de negocio para el programa de estudio.
- Matriz en la que se muestra la trazabilidad entre los resultados de negocio y los objetivos de aprendizaje.
- Resumen de este programa de estudio.

[1] **Consultar ****

0.3 Objetivos de Aprendizaje y Niveles de Conocimiento Evaluables

Los objetivos de aprendizaje son la base para los resultados de negocio y se utilizan para crear los exámenes de Probador Certificado del ISTQB® de Nivel Básico.

En general, todos los contenidos de este programa de estudio son evaluables en el nivel K1, excepto la Introducción y los Apéndices. Es decir, se le puede pedir al candidato que reconozca o recuerde 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 principio de cada capítulo y se clasifican de la siguiente manera:

- K1: recordar.
- K2: entender.
- K3: aplicar.

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

Las definiciones de todos los términos identificados como palabras clave, enumerados a continuación de los títulos de los capítulos, deben ser recordadas (K1) aunque no se mencione de forma explícita en los objetivos de aprendizaje.

0.4 El Examen de Certificación

El examen para obtener el Certificado de Nivel Básico se basará en este programa de estudio. Las respuestas a las preguntas de examen pueden requerir el uso de material basado en más de una sección de este programa de estudio. Todas las secciones del programa de estudio son evaluables, excepto la Introducción y los Apéndices. Los estándares, libros y otros programas de estudio del ISTQB se incluyen como referencias, pero su contenido no es evaluable, salvo lo que se recoge en este programa de estudio a partir de dichos estándares, libros y otros programas de estudio del 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).

Estos exámenes se pueden realizar como parte de un curso de formación acreditado o de forma independiente (por ejemplo, en un centro de exámenes o en un examen público). No es necesario realizar un curso de formación acreditado para realizar el examen.

0.5 Acreditación

Un Comité Miembro del ISTQB puede acreditar a los proveedores de formación cuyo material didáctico siga este programa. Los proveedores de formación deben obtener las directrices para la acreditación del Comité Miembro o del organismo que realiza la acreditación. Un curso acreditado es reconocido como conforme a este programa de estudio, y se le permite tener un examen ISTQB como parte del curso.

0.6 Nivel de Detalle

El nivel de detalle de este programa de estudio permite la realización de cursos y exámenes consistentes a nivel internacional. Para lograr este objetivo, el programa de estudio se compone de:

- Objetivos de aprendizaje generales que describen el propósito del Nivel Básico.
- Una lista de términos que los alumnos deben ser capaces de recordar.
- Objetivos de aprendizaje para cada área de conocimiento, que describen el resultado, en términos de conocimiento, del aprendizaje que se desea alcanzar.
- Una descripción de los conceptos clave, incluyendo referencias a fuentes tales como bibliografía o estándares aceptados.

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

0.7 Organización de este Programa de Estudio

Hay seis capítulos con contenido que puede ser objeto de examen. La “Duración” a continuación del título de nivel superior para cada capítulo especifica el tiempo para el capítulo; no se proporciona el tiempo a niveles inferiores a los de capítulo. Para los cursos de formación acreditados, el programa de estudio requiere un mínimo de 16,75 horas lectivas, distribuidas en seis capítulos, de la siguiente manera:

- Capítulo 1: 175 minutos Fundamentos del Proceso de Prueba
- Capítulo 2: 100 minutos Pruebas a lo Largo del Ciclo de Vida del Desarrollo de Software
- Capítulo 3: 135 minutos Prueba Estática
- Capítulo 4: 330 minutos Técnicas de Prueba
- Capítulo 5: 225 minutos Gestión del Proceso de Prueba
- Capítulo 6: 40 minutos Soporte de Herramientas para el Proceso de Prueba

1 Fundamentos del Proceso de Prueba

Duración: 175 minutos

Palabras Clave

análisis de prueba ("test analysis")
aseguramiento de la calidad ("quality assurance")
base de prueba ("test basis")
calendario de ejecución de prueba ("test execution schedule")
calidad ("quality")
caso de prueba ("test case")
causa raíz ("root cause")
cobertura ("coverage")
compleción de la prueba ("test completion")
condición de prueba ("test condition")
control de la prueba ("test control")
datos de prueba ("test data")
defecto ("defect")
depuración ("debugging")
diseño de la prueba ("test design")
ejecución de prueba ("test execution")
error ("error")
fallo ("failure")
implementación de prueba ("test implementation")
juego de prueba ("test suite")
monitorización de la prueba ("test monitoring")
objetivo de prueba ("test objective")
objeto de prueba ("test object")
oráculo de prueba ("test oracle")
planificación de prueba ("test planning")
procedimiento de prueba ("test procedure")
producto de prueba ("testware")
prueba ("testing")
trazabilidad ("traceability")
validación ("validation")
verificación ("verification)

Objetivos de Aprendizaje para Fundamentos de la Prueba:

1.1 ¿Qué es Probar?

NB-1.1.1 (K1) Identificar los objetivos característicos de la prueba.

NB-1.1.2 (K2) Diferenciar la prueba de la depuración.

1.2 ¿Por qué es Necesario Probar?

NB-1.2.1 (K2) Dar ejemplos de por qué es necesario probar.

NB-1.2.2 (K2) Describir la relación entre la prueba y el aseguramiento de la calidad y dar ejemplos de cómo probar contribuye a obtener un mayor nivel de calidad.

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

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

1.3 Siete Principios de la Prueba

NB-1.3.1 (K2) Explicar los siete principios del proceso de prueba.

1.4 Proceso de Prueba

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

NB-1.4.2 (K2) Describir las actividades de prueba y las tareas correspondientes dentro del proceso de prueba.

NB-1.4.3 (K2) Diferenciar los productos de trabajo que dan soporte al proceso de prueba.

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

1.5 La Psicología de la Prueba

NB-1.5.1 (K1) Identificar los factores psicológicos que influyen en el éxito de la prueba.

NB-1.5.2 (K2) Explicar la diferencia entre la forma de pensar requerida para las actividades de prueba y la forma de pensar requerida para las actividades de desarrollo.

1.1 ¿Qué es Probar?

Los sistemas software son una parte integral de la vida, desde las aplicaciones de negocio (por ejemplo, la banca) hasta los productos de consumo (por ejemplo, los automóviles). La mayoría de las personas ha tenido una experiencia con algún producto software que no funcionó como se esperaba. Un producto software que no funciona correctamente puede causar muchos problemas, incluyendo pérdida de dinero, tiempo o reputación en el ámbito del negocio, e incluso lesiones o muerte. Probar el software es una forma de evaluar la calidad del software y de reducir el riesgo de fallos en un entorno de operaciones o en producción.

Una percepción errónea común de la prueba es que sólo consiste en ejecutar pruebas, es decir, ejecutar el software y comprobar los resultados. Como se describe en la sección 1.4, la prueba de software es un proceso que incluye muchas actividades diferentes; la ejecución de la prueba (incluida la comprobación de los resultados) es sólo una de estas actividades. El proceso de prueba también incluye actividades tales como planificar la prueba, analizar, diseñar e implementar pruebas, informar del avance y de los resultados de la prueba, y evaluar la calidad de un objeto de prueba.

Alguna forma de prueba implica la ejecución del componente o sistema que se está probando; esta prueba se denomina prueba dinámica. Otras formas de prueba no implican la ejecución del componente o sistema que se está probando; estas formas de pruebas se denominan 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 común de la prueba es que se centra exclusivamente en la verificación de requisitos, historias de usuarios u otras especificaciones. Si bien las pruebas implican la comprobación de si el sistema cumple los requisitos especificados, también implican la validación, que consiste en comprobar si el sistema es capaz de satisfacer las necesidades de los usuarios y de otros implicados en su(s) entorno(s) de operación.

Las actividades de prueba se organizan y se llevan a cabo de forma diferente en diferentes ciclos de vida (véase el apartado 2.1).

1.1.1 Objetivos Característicos de la Prueba

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

- Evaluar productos de trabajo tales como requisitos, historias de usuario, diseño y código.
- Verificar el cumplimiento de todos los requisitos especificados.
- Validar si el objeto de prueba está completo y funciona como los usuarios y otros implicados esperan.
- Generar confianza en el nivel de calidad del objeto de prueba.
- Prevenir defectos.
- Encontrar fallos y defectos.
- Proporcionar suficiente información a los implicados para que puedan tomar decisiones informadas[2], especialmente en relación con el nivel de calidad del objeto de prueba.

[2] **Consultar ****

- Reducir el nivel de riesgo de calidad inadecuada del software (por ejemplo, fallos que se producen durante la operación que no han sido detectados anteriormente).
- Cumplir con requisitos o normas contractuales, legales o reglamentarias, y/o verificar el cumplimiento de dichos requisitos o normas por parte del objeto de prueba.

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

- Durante la prueba de componente, uno de los objetivos puede ser encontrar tantos fallos como sea posible para que los defectos subyacentes se identifiquen y se solucionen de forma temprana. Otro objetivo puede ser aumentar la cobertura de código de las pruebas de componente.
- Durante la prueba de aceptación, uno de los objetivos puede ser confirmar que el sistema funciona como se espera y cumple con los requisitos. Otro objetivo de este proceso de prueba puede ser informar a los implicados sobre el riesgo de liberar[3] el sistema en un momento dado.

1.1.2 Prueba y Depuración

La prueba y la depuración son diferentes. La ejecución de pruebas puede mostrar fallos causados por defectos en el software. La depuración es la actividad de desarrollo que encuentra, analiza y corrige dichos defectos. La prueba de confirmación posterior comprueba si las correcciones han resuelto 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 componente asociada. Sin embargo, en el desarrollo Ágil y en algunos otros ciclos de vida, los probadores pueden estar involucrados en la depuración y la prueba de componente.

La norma ISO (ISO/IEC/IEEE 29119-1) contiene información adicional sobre conceptos de la prueba de software.

1.2 ¿Por qué es Necesario Probar?

La prueba rigurosa de componentes y sistemas, y su documentación asociada, pueden ayudar a reducir el riesgo de que se produzcan fallos durante la operación. Cuando se detectan defectos, y posteriormente se corrigen[4], esto contribuye a la calidad de los componentes o sistemas. Además, la prueba del software también puede ser necesaria para cumplir con requisitos contractuales o legales o estándares específicos de la industria.

1.2.1 Contribuciones de la Prueba al Éxito

A lo largo de la historia de la informática, es bastante común que el software y los sistemas se entreguen a operaciones[5] y, debido a la presencia de defectos, que posteriormente causen fallos o no satisfagan, de algún otro modo, las necesidades de los implicados. Sin embargo, la utilización de técnicas de prueba adecuadas puede reducir la frecuencia de estas entregas problemáticas, cuando estas técnicas se aplican con el nivel adecuado de experiencia en materia de prueba, en los niveles de prueba adecuados y en los puntos adecuados del ciclo de vida del desarrollo del software. Algunos ejemplos son:

[3] **Consultar ****
[4] **Consultar ****
[5] **Consultar ****

- El hecho de contar con probadores involucrados en la revisión de los requisitos o en el refinamiento de historias de usuario podría resultar en la detección de defectos en estos productos de trabajo. La identificación y eliminación de defectos en los requisitos reduce el riesgo de que se desarrollen funcionalidades incorrectas o que no puedan ser probadas (ausencia de capacidad de ser probado).
- El hecho de que los probadores trabajen en estrecha colaboración con los diseñadores de sistemas mientras se diseña el sistema puede aumentar la comprensión de cada una de las partes sobre el diseño y la forma de probarlo. Esta mayor comprensión puede reducir el riesgo de defectos fundamentales de diseño y permitir la identificación de pruebas en una fase temprana.
- El hecho de que los probadores trabajen en estrecha colaboración con los desarrolladores mientras el código está en desarrollo puede aumentar la comprensión del código por parte de cada una de las partes y la forma de probarlo. Esta mayor comprensión puede reducir el riesgo de defectos dentro del código y de la prueba.
- Hacer que los probadores verifiquen y validen el software antes de liberarlo puede detectar fallos que de otro modo podrían haberse omitido, y apoyar el proceso de eliminación de los defectos que causaron los fallos (es decir, la depuración). Esto aumenta la probabilidad de que el software cumpla con las necesidades de los implicados y satisfaga los requisitos.

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

1.2.2 Aseguramiento de la Calidad y Proceso de Prueba

Mientras que, a menudo, las personas utilizan el término aseguramiento de la calidad[6] (o simplemente QA por sus siglas en inglés) para referirse a la prueba, el aseguramiento de la calidad y la prueba no son lo mismo, pero están relacionados. 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 se centra, por lo general, en el cumplimiento de los procesos adecuados, a fin de proporcionar la confianza de que se alcanzarán los niveles de calidad adecuados. Cuando los procesos se llevan a cabo de forma correcta, 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, son importantes para un aseguramiento de la calidad eficaz.

El control de la calidad implica varias actividades, incluyendo actividades de prueba, que apoyan el logro de niveles apropiados de calidad. Las actividades de prueba son parte del proceso general de desarrollo o mantenimiento del software. Dado que el aseguramiento de calidad se ocupa de la correcta ejecución de todo el proceso, el aseguramiento de calidad promueve la realización de la prueba adecuada. Como se describe en las secciones 1.1.1 y 1.2.1, la prueba contribuye a la consecución de la calidad de diversas maneras.

[6] **Consultar ****

1.2.4 Errores, Defectos y Fallos

Una persona puede cometer un error (equivocación), que puede llevar a la introducción de un defecto (falta o bug) en el código software o en algún otro producto de trabajo relacionado. Un error que conduce a la introducción de un defecto en un producto de trabajo puede desencadenar un error que conduce a la introducción de un defecto en un producto de trabajo con el que se encuentra relacionado. Por ejemplo, un error en la educción[7] de requisitos puede conducir a un defecto en el requisito, lo que a su vez resulta en un error de programación que conduce a un defecto en el código.

Si se ejecuta un fragmento de código que contiene un defecto, esto puede causar un fallo, pero no necesariamente en todas las circunstancias. Por ejemplo, algunos defectos requieren entradas o precondiciones muy específicas para desencadenar un fallo, que puede ocurrir rara vez o nunca.

Los errores pueden ocurrir por muchas razones, tales como:

- Presión por causa de tiempo.
- Falibilidad humana.
- Participantes en el proyecto sin experiencia o poco cualificados.
- Falta de comunicación entre los participantes en el proyecto, incluida la falta de comunicación con respecto a requisitos y diseño.
- Complejidad del código, diseño, arquitectura, el problema subyacente que se debe resolver, y/o las tecnologías utilizadas.
- Malentendidos acerca de las interfaces intra e intersistemas, especialmente cuando esas interacciones intra e intersistemas son numerosas.
- Tecnologías nuevas y desconocidas.

Además de los fallos causados por defectos en el código, los fallos también pueden ser causados por condiciones medioambientales. 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 la prueba son fallos. Pueden ocurrir falsos positivos 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 producto de prueba, o por otras razones. La situación inversa también puede ocurrir, 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 informan como defectos, pero en realidad no son defectos.

1.2.5 Defectos, Causas Raíz y Efectos

Las causas raíz de los defectos son las acciones o condiciones más tempranas que contribuyeron a crear estos defectos. Se pueden analizar los defectos para identificar sus causas raíz, con el propósito de reducir la ocurrencia de defectos similares en el futuro. Al centrarse en las causas raíz más significativas, el análisis de la causa raíz puede conducir a mejoras en el proceso que previenen la introducción de un número significativo de futuros defectos.

Por ejemplo, suponer que los pagos de intereses incorrectos, debido a una sola línea de código incorrecto, dan lugar a reclamaciones por parte de los clientes. El código defectuoso fue escrito para una historia de

[7] **Consultar ****

usuario que era ambigua, debido al malentendido del propietario del producto sobre cómo calcular el interés. Si existe un gran porcentaje de defectos en el cálculo de intereses, y estos defectos tienen su origen en malentendidos similares, los propietarios de producto podrían recibir formación en materia de cálculo de intereses para reducir dichos defectos en el futuro.

En este ejemplo, las reclamaciones de los clientes son los 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 resultó en que el propietario del producto cometiera una equivocación al escribir la historia de usuario. El proceso de análisis de la causa raíz se discute en el Programa de Estudio de Gestión de Pruebas de Nivel Experto (ISTQB-ETM por sus siglas en inglés) y en el Programa de Estudio de Mejora del Proceso de Pruebas de Nivel Experto (ISTQB-EITP por sus siglas en inglés) del ISTQB.

1.3 Siete Principios de la Prueba

En los últimos 50 años se han sugerido una serie de principios del proceso de prueba que ofrecen directrices generales comunes para toda prueba.

1. La prueba muestra la presencia de defectos, no su ausencia

La prueba puede mostrar la presencia de defectos, pero no puede probar que no hay defectos. La prueba reduce la probabilidad de que queden defectos no descubiertos en el software pero, incluso si no se encuentran defectos, el proceso de prueba no es una demostración de la corrección.

2. La prueba exhaustiva es imposible

No es posible probar todo (todas las combinaciones de entradas y precondiciones) excepto en casos triviales. En lugar de intentar realizar pruebas exhaustivas se deberían utilizar el análisis de riesgos, las técnicas de prueba y las prioridades para centrar los esfuerzos de prueba.

3. La prueba temprana ahorra tiempo y dinero

Para detectar defectos de forma temprana, las actividades de prueba tanto estáticas como dinámicas deben iniciarse lo antes posible en el ciclo de vida de desarrollo de software. La prueba temprana a veces se denomina *desplazamiento hacia la izquierda*[8]. La prueba temprana en el ciclo de vida de desarrollo de software ayuda a reducir o eliminar cambios costosos (ver sección 3.1).

4. Los defectos se agrupan

En general, un pequeño número de módulos contiene la mayoría de los defectos descubiertos durante la prueba previa al lanzamiento, o es responsable de la mayoría de los fallos operativos. Las agrupaciones de defectos previstas y las agrupaciones de defectos reales observadas en la prueba o producción son una aportación importante a un análisis de riesgos utilizado para centrar el esfuerzo de la prueba (como se menciona en el principio 2).

[8] **Consultar ****

5. Cuidado con la paradoja del pesticida

Si las mismas pruebas se repiten una y otra vez, eventualmente estas pruebas ya no encontrarán ningún defecto nuevo. Para detectar nuevos defectos, es posible que sea necesario cambiar las pruebas y los datos de prueba existentes, y es posible que sea necesario redactar nuevas pruebas. (Las pruebas ya no son efectivas para detectar defectos, de la misma manera que los pesticidas ya no son efectivos para matar insectos después de un tiempo). En algunos casos, como la prueba de regresión automatizada, la paradoja del pesticida tiene un resultado beneficioso, que es el número relativamente bajo de defectos de regresión.

6. La prueba depende del contexto

La prueba se realiza de manera diferente en diferentes contextos. Por ejemplo, el software de control industrial de seguridad crítica se prueba de forma diferente a una aplicación móvil de comercio electrónico. Como ejemplo adicional, la prueba en un proyecto Ágil se realiza de manera diferente a la prueba en un proyecto que se desarrolla según un ciclo de vida secuencial (ver 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 equivocada) esperar que sólo con encontrar y corregir un gran número de defectos se asegure el éxito de un sistema. Por ejemplo, la realización de pruebas exhaustivas de todos los requisitos especificados y la reparación de todos los defectos encontrados podrían dar lugar a un sistema difícil de utilizar, que no satisface las necesidades y expectativas de los usuarios o que es peor en comparación con otros sistemas de la competencia.

Consultar Myers 2011, Kaner 2002 y Weinberg 2008 para ejemplos de estos y otros principios del proceso de prueba.

1.4 Proceso de Prueba

No existe un proceso de prueba de software único y universal, pero existen conjuntos de actividades de prueba comunes sin las cuales es menos probable que la prueba alcance los objetivos establecidos. Estos conjuntos de actividades de prueba son un proceso de prueba. El proceso de prueba de software adecuado y específico en cualquier situación depende de muchos factores. Qué actividades de prueba están involucradas en este proceso de prueba, cómo se implementan estas actividades y cuándo ocurren, pueden ser abordadas en la estrategia de prueba de una organización.

1.4.1 El Proceso de Prueba en Contexto

Los factores de contexto que influyen en el proceso de prueba de una organización incluyen, pero no están limitados a:

- Modelo de ciclo de vida de desarrollo de software y metodologías de proyecto en uso.
- Niveles y tipos de prueba considerados.
- Riesgos de producto y de proyecto.
- Dominio del negocio.
- Restricciones operativas, incluyendo pero no limitadas a:
    - Presupuestos y recursos.

- Plazos.
- Complejidad.
- Requisitos contractuales y normativos.

- Políticas y prácticas de la organización.
- Estándares internos y externos necesarios.

Las siguientes secciones describen los aspectos generales de los procesos de prueba en una organización en los siguientes términos:

- Actividades y tareas de prueba.
- Productos de trabajo de la prueba.
- Trazabilidad entre la base de prueba y los productos de trabajo de la prueba.

Es muy útil si la base de prueba (para cualquier nivel o tipo de prueba que se esté considerando) tiene definidos criterios de cobertura medibles. Los criterios de cobertura pueden actuar eficazmente como indicadores clave de desempeño[9] (KPI's[10], por sus siglas en inglés) para controlar las actividades que permiten demostrar el logro de los objetivos de la prueba de software (ver 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 soportados. Cada requisito es un elemento de la base de prueba. Cada dispositivo soportado es también un elemento de la base de prueba. Es posible que los criterios de cobertura requieran, como mínimo, un caso de prueba para cada elemento de la base de prueba. Una vez ejecutadas, los resultados de estas pruebas indican a los implicados si se cumplen los requisitos especificados y si se observaron fallos en los dispositivos soportados.

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

1.4.2 Actividades y Tareas de Prueba

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

- Planificación de la prueba.
- Monitorización y control de la prueba (también seguimiento y control de la prueba).
- Análisis de la prueba.
- Diseño de la prueba.
- Implementación de la prueba.
- Ejecución de la prueba.
- Compleción[11] de la prueba.

Cada grupo de actividades se compone de actividades integrantes, que se describirán en las siguientes subsecciones. Cada actividad dentro de cada grupo de actividades a su vez puede estar compuesta por múltiples tareas individuales, que pueden variar de un proyecto a otro o de un lanzamiento a otro.

[9] **Consultar ****
[10] **Consultar ****
[11] **Consultar ****

Además, aunque muchos de estos grupos de actividades pueden parecer secuenciales, desde un punto de vista lógico, a menudo se implementan de manera iterativa. Por ejemplo, el desarrollo Ágil implica pequeñas iteraciones de diseño, construcción y prueba de software que se realizan de forma continua, con el apoyo de una planificación continua. Por lo tanto, las actividades de prueba también se llevan a cabo 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 normalmente es necesario adaptar estas actividades principales dentro del contexto del sistema y del proyecto.

Planificación de la Prueba

La planificación de la prueba implica actividades que definen los objetivos de la prueba y el enfoque para cumplir con los objetivos de la prueba dentro de las restricciones impuestas por el contexto (por ejemplo, la especificación de técnicas y tareas de prueba adecuadas y la formulación de un calendario de pruebas para cumplir con un plazo límite). Los planes de prueba pueden ser revisados en función de la retroalimentación 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 (o seguimiento) de la prueba implica la comparación continua del avance real con respecto al plan de prueba utilizando cualquier métrica de monitorización de la prueba definida en el plan de prueba. El control de la prueba implica tomar las medidas necesarias para cumplir los objetivos del plan de prueba (que puede actualizarse con el tiempo). La monitorización y el control de la prueba se apoyan en la evaluación de los criterios de salida, a los que se hace referencia como definición de hecho[12] en algunos ciclos de vida (ver Programa de Estudio de Probador Certificado del ISTQB - Nivel Básico - Extensión Ágil - 2014). 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:

- Comprobar los resultados y los registros de la prueba en relación con los criterios de cobertura especificados.
- Evaluar el nivel de calidad de los componentes o sistemas en base a los resultados y los registros de prueba.
- Determinar si se necesitan más pruebas (por ejemplo, si las pruebas originalmente destinadas a alcanzar un cierto nivel de cobertura de riesgo de producto no lo alcanzaran, sería necesario redactar y ejecutar pruebas adicionales).

El avance de la prueba con respecto al plan se comunica a los implicados por medio de informes de avance de la prueba, incluyendo las desviaciones con respecto al plan y la información que permita apoyar cualquier decisión de interrumpir la prueba.

La monitorización y el control de la prueba se explican con más detalle en la sección 5.3.

Análisis de la Prueba

Durante el análisis de la prueba, se analiza la base de prueba para identificar las prestaciones[13] que presentan capacidad de ser probadas y definir las condiciones de prueba asociadas. En otras palabras, el análisis de la prueba determina "qué probar" en términos de criterios de cobertura medibles.

[12] **Consultar ****
[13] **Consultar ****

El análisis de la prueba incluye las siguientes actividades principales:

- Analizar la base de prueba correspondiente al nivel de prueba considerado, por ejemplo:
  - Especificaciones de requisitos, tales como requisitos de negocio, requisitos funcionales, requisitos de sistema, historias de usuario, épicas, casos de uso o productos de trabajo similares que especifiquen el comportamiento funcional y no funcional deseado del componente o sistema.
  - Información de diseño e implementación, tales como diagramas o documentos de arquitectura del sistema o del software, especificaciones de diseño, flujos de llamada, diagramas de modelado (por ejemplo, diagramas UML o entidad-relación), especificaciones de interfaz o productos de trabajo similares que especifiquen componentes o estructura del sistema.
  - La implementación del componente o sistema en sí, incluyendo código, metadatos de base de datos y consultas, e 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 distintos tipos, tales como:
  - Ambigüedades.
  - Omisiones.
  - Inconsistencias.
  - Inexactitudes.
  - Contradicciones.
  - Enunciados superfluos.
- Identificar las prestaciones y conjuntos de prestaciones que se van a probar.
- Definir y priorizar las condiciones de prueba para cada prestación basándose en el análisis de la base de prueba, y considerando las características funcionales, no funcionales y estructurales, otros factores de negocio 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 (véanse las 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 la prueba (véase el 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 la prueba produce condiciones de prueba que deben ser utilizadas como objetivos de prueba en los contratos de prueba. Los contratos de prueba son productos de trabajo característicos en algunos tipos de prueba basadas en la experiencia (véase la sección 4.4.2). Cuando estos objetivos de prueba son trazables a la base de prueba, se puede medir la cobertura alcanzada durante dicha prueba basada en la experiencia.

La identificación de defectos durante el análisis de la prueba es un beneficio potencial importante, especialmente cuando no se utiliza ningún otro proceso de revisión y/o cuando el proceso de prueba está estrechamente vinculado al mismo proceso de revisión. Estas actividades de análisis de prueba no sólo verifican si los requisitos son consistentes, están debidamente expresados y son completos, sino que

también validan si los requisitos satisfacen adecuadamente las necesidades de los clientes, usuarios y otros implicados. Por ejemplo, técnicas como el desarrollo dirigido por el comportamiento (BDD por sus siglas en inglés) y el desarrollo dirigido por prueba de aceptación (ATDD por sus siglas en inglés), que involucran la generación de condiciones de prueba y casos de prueba a partir de historias de usuarios y criterios de aceptación antes de la codificación, también verifican, validan y detectan defectos en las historias de usuarios y en los criterios de aceptación (ver el Programa de Estudio de Probador Certificado del ISTQB - Nivel Básico - Extensión Ágil - 2014).

Diseño de la Prueba

Durante el diseño de la prueba, las condiciones de prueba se transforman en casos de prueba de alto nivel, conjuntos de casos de prueba de alto nivel y otros productos de prueba. Es decir, el análisis de la prueba responde a la pregunta "qué probar" y el diseño de la prueba responde a la pregunta "cómo probar".

El diseño de la prueba incluye las siguientes actividades principales:

- Diseñar y priorizar casos de prueba y conjuntos de casos de prueba.
- Identificar los datos de prueba necesarios para apoyar las condiciones de prueba y los casos de prueba.
- Diseñar el entorno de prueba e identificar la infraestructura y las herramientas necesarias.
- Capturar la trazabilidad bidireccional entre la base de prueba, las condiciones de prueba, los casos de prueba y los procedimientos de prueba (véase la sección 1.4.4).

El desarrollo, a partir de condiciones de prueba, de casos de prueba y de conjuntos de casos de prueba durante el diseño de la prueba implica, a menudo, el uso de técnicas de prueba (véase el capítulo 4).

Al igual que con el análisis de la prueba, el diseño de la prueba también puede dar lugar a la identificación de tipos similares de defectos en la base de prueba. Al igual que con el análisis de la prueba, la identificación de defectos durante el diseño de la prueba es un beneficio potencial importante.

Implementación de la Prueba

Durante la implementación de la prueba, se crean y/o se completan los productos de prueba necesarios para la ejecución de la prueba, incluyendo la secuenciación de los casos de prueba en procedimientos de prueba. Por lo tanto, el diseño de la prueba responde a la pregunta "cómo probar", mientras que la implementación de la prueba responde a la pregunta " ¿está todo preparado para realizar la prueba?

La implementación de la prueba incluye las siguientes actividades principales:

- Desarrollar y priorizar procedimientos de prueba y, eventualmente, crear guiones de prueba automatizados.
- Crear juegos de prueba a partir de los procedimientos de prueba y (si los hubiera) guiones de prueba automatizados.
- Organizar los juegos de prueba dentro de un calendario de ejecución de la prueba de forma que se obtenga una ejecución eficiente de los mismos (véase el apartado 5.2.4).
- Construir el entorno de prueba (incluyendo, posiblemente, arneses de prueba, virtualización de servicios, simuladores y otros elementos de infraestructura) y verificar que se haya configurado correctamente todo lo necesario.
- Preparar los datos de prueba y asegurarse de que estén correctamente cargados 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 (véase la sección 1.4.4).

A menudo se combinan las tareas de diseño y ejecución de la prueba.

Es posible que en la prueba exploratoria y otros tipos de prueba basada en la experiencia se desarrollen el diseño y la implementación de la prueba, que pueden ser documentados como parte de la ejecución de la prueba. La prueba exploratoria puede basarse en contratos de prueba (producidos como parte del análisis de la prueba), y la prueba exploratoria se ejecuta inmediatamente, a medida que se diseña e implementa (véase la sección 4.4.2).

Ejecución de la Prueba

Durante la ejecución de la prueba, los juegos de prueba se ejecutan de acuerdo al calendario de ejecución de la prueba.

La ejecución de la prueba incluye las siguientes actividades principales:

- Registrar los identificadores y las versiones de los elementos u objetos de prueba, las herramientas de prueba y los productos de prueba.
- Ejecutar pruebas de forma manual o utilizando herramientas de ejecución de pruebas.
- Comparar los resultados reales con los resultados esperados.
- Analizar las anomalías para establecer sus causas probables (por ejemplo, pueden ocurrir fallos debido a defectos en el código, pero también pueden ocurrir falsos positivos [ver sección 1.2.3]).
- Informar sobre los defectos en función de los fallos observados (véase el apartado 5.6).
- Registrar el resultado de la ejecución de la prueba (por ejemplo, pasada, fallada, bloqueada).
- Repetir las actividades de prueba ya sea como resultado de una acción tomada para una anomalía, o como parte de la prueba planificada (por ejemplo, la ejecución de una prueba corregida, una prueba de confirmación y/o una prueba de regresión).
- Verificación y actualización de la trazabilidad bidireccional entre la base de prueba, las condiciones de prueba, los casos de prueba, los procedimientos de prueba y los resultados de la prueba.

Compleción de la Prueba

Las actividades de compleción de la prueba recopilan datos de las actividades de prueba completadas para consolidar la experiencia, los productos de prueba y cualquier otra información relevante. Las actividades de compleción de la prueba ocurren en hitos del proyecto tales como cuando un sistema software es liberado, un proyecto de prueba es completado (o cancelado), cuando finaliza una iteración de un proyecto Ágil (por ejemplo, como parte de una reunión retrospectiva), cuando se completa un nivel de prueba, o cuando se completa la liberación de un mantenimiento.

La compleción de la prueba incluye las siguientes actividades principales:

- Comprobar que todos los informes de defecto estén cerrados, registrando solicitudes de cambio o elementos de la cartera del producto[14] para cualquier defecto que quede sin resolver al final de la ejecución de la prueba. Crear un informe resumen de prueba que se comunicará a los implicados.
- Finalizar, archivar y almacenar (según corresponda) el entorno de prueba, los datos de prueba, la infraestructura de prueba y otros productos de prueba para su posterior reutilización.
- Traspaso de los productos de prueba a los equipos de mantenimiento, a otros equipos del proyecto y/u otros implicados que podrían beneficiarse de su uso.
- Analizar las lecciones aprendidas de las actividades de prueba completadas para determinar los cambios necesarios para iteraciones, lanzamientos y proyectos futuros.
- Utilizar la información recopilada para mejorar la madurez del proceso de prueba.

1.4.3 Productos de Trabajo de la Prueba

Los productos de trabajo de la prueba se crean como parte del proceso de prueba. Así como hay una variación significativa en la forma en que las organizaciones implementan el proceso de prueba, también hay una variación significativa en los tipos de productos de trabajo creados durante ese proceso, en las formas en que esos productos de trabajo están organizados y gestionados, y en los nombres que se utilizan para esos productos de trabajo. Este programa de estudio se adhiere al proceso de prueba descrito anteriormente y a los productos de trabajo descritos en este programa de estudio y en el Glosario del ISTQB. La norma ISO (ISO/IEC/IEEE 29119-3) también puede servir como guía para los productos de trabajo de la prueba.

Muchos de los productos de trabajo de la prueba descritos en esta sección se pueden capturar y gestionar utilizando herramientas de gestión de pruebas y herramientas de gestión de defectos (véase el capítulo 6).

Productos de Trabajo de la Planificación de la Prueba

Los productos de trabajo de la planificación de la prueba, por lo general, incluyen uno o más planes de prueba. El plan de prueba incluye información sobre la base de prueba, con la que se relacionarán los demás productos de trabajo de la prueba mediante información de trazabilidad (véase más adelante y el apartado 1.4.4), así como los criterios de salida (o definición de hecho) que se utilizarán durante la monitorización y el control de la prueba. Los planes de prueba se describen en la sección 5.2.

Productos de Trabajo de la Monitorización y Control de la Prueba

Los productos de trabajo de la monitorización y control de la prueba, normalmente, incluyen varios tipos de informes de prueba, incluyendo informes del avance de la prueba (producidos de forma continua y/o regular) e informes resumen 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 la fecha del informe, incluyendo un resumen de los resultados de la ejecución de la prueba una vez que estén disponibles.

Los productos de trabajo de monitorización y control de la prueba también deben abordar las inquietudes de la gestión del proyecto, tales como la compleción de tareas, la asignación y el uso de recursos, y el esfuerzo.

[14] **Consultar ****

La monitorización y el control de la prueba, así como los productos de trabajo creados durante estas actividades, se explican con más detalle en la sección 5.3 de este programa de estudio.

Productos de Trabajo del Análisis de la Prueba

Los productos de trabajo del análisis de la prueba incluyen condiciones de prueba definidas y priorizadas, cada una de las cuales es, idealmente, trazable bidireccionalmente a los elementos específicos de la base de prueba que cubre. Para la prueba exploratoria, el análisis de la prueba puede implicar la creación de contratos de prueba. El análisis de la prueba también puede dar lugar al descubrimiento y notificación de defectos en la base de prueba.

Productos de Trabajo del Diseño de la Prueba

El diseño de la prueba resulta en casos de prueba y conjuntos de casos de prueba para practicar[15] las condiciones de prueba definidas en el análisis de la 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, sin dejar de documentar adecuadamente el alcance del caso de prueba. Lo ideal es que cada caso de prueba se pueda trazar bidireccionalmente hasta la(s) condición(es) de prueba que cubre.

El diseño de la prueba también da lugar al 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 la prueba pueden refinarse aún más en el diseño de la prueba.

Productos de Trabajo de la Implementación de la Prueba

Los productos de trabajo de la implementación de la prueba incluyen:

- Procedimientos de prueba y la secuenciación de dichos procedimientos de prueba.
- Juegos de prueba.
- Un calendario de ejecución de pruebas.

Idealmente, una vez finalizada la implementación de la prueba, el logro de los criterios de cobertura establecidos en el plan de prueba puede demostrarse mediante 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 la prueba implica la creación de productos de trabajo que utilizan o son utilizados por herramientas, como la virtualización de servicios y los guiones de prueba automatizados.

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

[15] **Consultar ****

Los datos de prueba sirven para asignar valores concretos a las entradas y a 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 utilizar diferentes datos de prueba cuando se ejecuta en diferentes versiones del objeto de prueba. Los resultados esperados concretos que se asocian con los datos de prueba concretos se identifican mediante el uso de un oráculo de prueba.

En la prueba exploratoria, se pueden crear algunos productos de trabajo de diseño e implementación de la prueba durante la ejecución de la misma, aunque la medida en que se documenta la prueba exploratoria (y su trazabilidad a elementos específicos de la base de prueba) puede variar significativamente.

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

Productos de Trabajo de la Ejecución de la Prueba

Los productos de trabajo de la ejecución de la prueba incluyen:

- Documentación sobre el estado de casos de prueba individuales o procedimientos de prueba (por ejemplo, listo para ejecutar, pasado, fallado, bloqueado, omitido de forma deliberada, etc.).
- Informes de defecto (vea el párrafo 5.6).
- Documentación sobre el o los elemento(s), objeto(s), herramienta(s) y productos de prueba[16] involucrados en la prueba.

Idealmente, una vez finalizada la ejecución de la prueba, se puede determinar e informar el estado de cada elemento de la base de prueba a través de la trazabilidad bidireccional con el procedimiento o procedimientos de prueba asociados. Por ejemplo, se puede decir qué requisitos han pasado todas las pruebas planificadas, qué requisitos han fallado y/o tienen defectos asociados a ellos, y qué requisitos tienen pruebas planificadas que se encuentran a la espera de ser ejecutadas. Esto permite verificar que se hayan cumplido los criterios de cobertura, y permite informar los resultados de la prueba en términos comprensibles para todos los implicados.

Productos de Trabajo de la Compleción de la Prueba

Los productos de trabajo de la compleción de la prueba incluyen informes de resumen de la prueba, elementos de acción para la mejora de proyectos o iteraciones subsiguientes (por ejemplo, siguiendo una retrospectiva ágil del proyecto), solicitudes de cambio o elementos de la cartera del producto, y productos de prueba finalizados.

[16] **Consultar ****

1.4.5 Trazabilidad entre la Base de Prueba y los Productos de Trabajo de la Prueba

Como se mencionó en la sección 1.4.3, los productos de trabajo de la prueba y los nombres de esos productos de trabajo varían de forma significativa. Independientemente de estas variaciones, para implementar una monitorización y control efectivos de la prueba, 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 la prueba asociados con ese elemento, tal como se describió anteriormente. Además de la evaluación de la cobertura de las pruebas, una buena trazabilidad permite:

- Analizar el impacto de cambios.
- Hacer que la prueba pueda ser auditada.
- Cumplimiento de los criterios de gobernanza de TI.
- Mejorar la comprensión de los informes de avance de la prueba y de los informes resumen de prueba para incluir el estado de los elementos de la base de prueba (por ejemplo, los requisitos que pasaron sus pruebas, los requisitos que fallaron sus pruebas, y los requisitos que tienen pruebas pendientes).
- Relacionar los aspectos técnicos de la prueba con los implicados en términos que éstos puedan entender.
- Aportar información para evaluar la calidad de los productos, la capacidad de los procesos y el avance de los proyectos en relación con los objetivos de negocio.

Algunas herramientas de gestión de la prueba proporcionan modelos de productos de trabajo de la prueba que coinciden con parte o la totalidad de los productos de trabajo de la 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 del Proceso de Prueba

El desarrollo de software, incluyendo la prueba de software, involucra a seres humanos. Por lo tanto, la psicología humana tiene efectos importantes en la prueba de software.

1.5.1 Psicología Humana y el Proceso de Prueba

La identificación de defectos durante una prueba estática, como una revisión de requisitos o una sesión de refinamiento de historias de usuario, o la identificación de fallos durante la ejecución de una prueba dinámica, puede ser percibida 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 esté en desacuerdo con las creencias actuales. Por ejemplo, dado que los desarrolladores esperan que su código sea correcto, tienen un sesgo de confirmación que hace difícil aceptar que el código sea incorrecto. Además del sesgo de confirmación, otros sesgos cognitivos pueden dificultar que las personas entiendan o acepten la información producida por la prueba. Asimismo, es un rasgo humano común culpar al portador de malas noticias, y la información producida por la prueba, a menudo, contiene malas noticias.

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

Los probadores y jefes de prueba necesitan tener buenas competencias[17] interpersonales para poder comunicarse eficazmente sobre defectos, fallos, resultados de la prueba, avance de la prueba y riesgos, y para construir relaciones positivas con las personas que forman parte de su entorno de trabajo. Las formas de comunicarse de manera adecuada incluyen los siguientes ejemplos:

- Comenzar con colaboración en lugar de batallas. Recordar a todos que el objetivo común es lograr sistemas de mejor calidad.
- Enfatizar los beneficios de la prueba. Por ejemplo, para los autores, la información sobre los defectos puede ayudarles a mejorar sus productos de trabajo y sus competencias. Para la organización, los defectos detectados y corregidos durante la prueba ahorrarán tiempo y dinero y reducirán el riesgo general para la calidad de producto.
- Comunicar los resultados de las pruebas y otros hallazgos de manera neutral y centrada en los hechos sin criticar a la persona que creó el elemento defectuoso. Redactar informes de defecto objetivos y fundamentados en hechos y revisar los hallazgos.
- Tratar de entender cómo se siente la otra persona y las razones por las que puede reaccionar negativamente a la información.
- Confirmar que el interlocutor ha entendido lo que se ha dicho y viceversa.

Los objetivos característicos de la prueba se discutieron anteriormente (ver sección 1.1). Definir claramente 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 dirección y otros implicados. También es importante que los probadores se adhieran a estos objetivos con un mínimo sesgo personal.

1.5.2 Formas de Pensar del Probador y del Desarrollador

Los desarrolladores y los probadores, a menudo, piensan de forma diferente. El objetivo principal del desarrollo es diseñar y construir un producto. Como se expuso anteriormente, los objetivos de la prueba incluyen la verificación y validación del producto, la detección de defectos antes de liberarlo, y así sucesivamente. Se trata de diferentes conjuntos de objetivos que requieren diferentes mentalidades, o formas de pensar. La combinación de estas mentalidades ayuda a lograr un mayor nivel de calidad del producto.

Una mentalidad refleja las suposiciones de un individuo y los métodos preferidos para la toma de decisiones y la resolución de problemas. La mentalidad de un probador debe incluir curiosidad, pesimismo profesional, sentido crítico, atención al detalle y motivación para una comunicación y relaciones buenas y positivas. La mentalidad de un probador tiende a ampliarse y madurar a medida que adquiere experiencia.

La mentalidad de un desarrollador puede incluir algunos de los elementos de la mentalidad de un probador, pero los desarrolladores de éxito, a menudo, están más interesados en el diseño y la construcción de soluciones que en la contemplación de lo que podría estar mal en esas soluciones. Además, el sesgo de confirmación hace difícil encontrar errores en su propio trabajo.

Con la mentalidad correcta, los desarrolladores pueden probar su propio código. Los diferentes modelos de ciclo de vida de desarrollo de software, a menudo, tienen diferentes maneras de organizar a los probadores y las actividades de prueba. El hecho de que algunas de las actividades de prueba sean realizadas por probadores independientes aumenta la eficacia de la detección de defectos, lo cual es particularmente importante para sistemas grandes, complejos o de seguridad crítica. Los probadores

[17] **Consultar ****

independientes aportan una perspectiva diferente a la de los autores de los productos de trabajo (es decir, analistas de negocio, propietarios de producto, diseñadores y programadores), ya que tienen diferentes sesgos cognitivos de los autores.

2 Prueba a lo Largo del Ciclo de Vida de Desarrollo de Software

Duración: 100 minutos

Palabras Clave

análisis de impacto ("impact analysis")
base de prueba ("test basis")
caso de prueba ("test case")
entorno de prueba ("test environment")
modelo de desarrollo secuencial ("sequential development model")
nivel de prueba ("test level")
objetivo de prueba ("test objective")
objeto de prueba ("test object")
prueba alfa ("alpha testing")
prueba beta ("beta testing")
prueba de aceptación ("acceptance testing")
prueba de aceptación contractual ("contractual acceptance testing")
prueba de aceptación de usuario ("user acceptance testing")
prueba de aceptación operativa ("operational acceptance testing")
prueba de aceptación regulatoria ("regulatory acceptance testing")
prueba de caja blanca ("white-box testing")
prueba de componente ("component testing")
prueba de confirmación ("confirmation testing")
prueba de integración ("integration testing")
prueba de integración de componente ("component integration testing")
prueba de integración de sistemas ("system integration testing")
prueba de mantenimiento ("maintenance testing")
prueba de regresión ("regression testing")
prueba de sistema ("system testing")
prueba funcional ("functional testing")
prueba no funcional ("non-functional testing")
software comercial de distribución masiva ("commercial off-the-shelf (COTS)")
tipo de prueba ("test type")

Objetivos de Aprendizaje para Prueba a lo Largo del Ciclo de Vida del Desarrollo de Software

2.1 Modelos de Ciclo de Vida del Desarrollo de Software

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

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

2.2 Niveles de Prueba

NB-2.2.1 (K2) Comparar los diferentes niveles de prueba desde la perspectiva de los objetivos, bases de prueba, objetos de prueba, defectos y fallos característicos, y enfoques y responsabilidades.

2.3 Tipos de Prueba

NB-2.3.1 (K2) Comparar los tipos de prueba funcional, no funcional y de caja blanca.

NB-2.3.2 (K1) Reconocer que los tipos de prueba funcional, no funcional y de caja blanca tienen lugar en cualquier nivel de prueba.

NB-2.3.3 (K2) Comparar los objetivos de la prueba de confirmación y la prueba de regresión.

2.4 Prueba de Mantenimiento

NB-2.4.1 (K2) Recapitular los desencadenantes de la prueba de mantenimiento.

NB-2.4.2 (K2) Describir el papel del análisis de impacto en la prueba de mantenimiento.

2.1 Modelos de Ciclo de Vida del 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 forma lógica y cronológica. Hay diferentes modelos de ciclo de vida de desarrollo de software, cada uno de los cuales requiere diferentes enfoques de prueba.

2.1.1 Desarrollo de Software y Prueba de Software

Una parte importante del rol de probador es estar familiarizado con los modelos de ciclo de vida de desarrollo de software más comunes, de modo que se puedan realizar las actividades de prueba adecuadas.

En cualquier modelo de ciclo de vida de desarrollo de software hay una serie de características que hacen que las pruebas sean adecuadas:

- Para cada actividad de desarrollo, hay una actividad de prueba asociada.
- 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 comienza durante la actividad de desarrollo correspondiente.
- Los probadores participan en discusiones para definir y refinar los requisitos y el diseño, y están involucrados en la revisión de los productos de trabajo (por ejemplo, requisitos, diseño, historias de usuario, etc.) tan pronto como las versiones borrador estén disponibles.

Independientemente del modelo de ciclo de vida de desarrollo de software que se haya elegido, las actividades de prueba deben comenzar en las etapas iniciales del ciclo de vida, adhiriéndose al principio de “prueba temprana”.

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

- Modelos de desarrollo secuencial.
- Modelos de desarrollo iterativos e incrementales.

Un modelo de desarrollo secuencial describe el proceso de desarrollo de software como un flujo lineal y secuencial de actividades. Esto significa que cualquier fase del proceso de desarrollo debe comenzar cuando se haya completado la fase anterior. En teoría, no hay solapamiento de fases, pero en la práctica, es beneficioso tener una retroalimentación temprana de la fase siguiente.

En el modelo en Cascada, las actividades de desarrollo (por ejemplo, análisis de requisitos, diseño, codificación, prueba) se completan una tras otra. En este modelo, las actividades de prueba sólo ocurren después de que todas las demás actividades de desarrollo hayan sido completadas.

A diferencia del modelo en Cascada, el modelo en V integra el proceso de prueba a lo largo de todo el proceso de desarrollo, implementando el principio de la prueba temprana. Además, el modelo en V incluye niveles de prueba asociados con cada fase de desarrollo correspondiente, lo que favorece aún más la prueba temprana (véase la sección 2.2 para un análisis de los niveles de prueba). En este modelo, la ejecución de las pruebas asociadas a cada nivel de prueba tiene lugar de forma secuencial, pero en algunos casos se producen solapamientos.

Los modelos de desarrollo secuencial ofrecen software que contiene el conjunto completo de prestaciones, pero normalmente requieren meses o años para su entrega a los implicados y usuarios.

El desarrollo incremental implica establecer requisitos, diseñar, construir y probar un sistema en fragmentos, lo que significa que las prestaciones del software crecen de forma incremental. El tamaño de estos incrementos de prestaciones varía, ya que algunos métodos tienen piezas más grandes y otros más pequeñas. Los incrementos de prestaciones pueden ser tan pequeños como un simple cambio en una pantalla de la interfaz de usuario o una nueva opción en una consulta.

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

Algunos ejemplos son:

- Rational Unified Process: Cada iteración tiende a ser relativamente larga (por ejemplo, de dos a tres meses), y los incrementos de las prestaciones son proporcionalmente grandes, como por ejemplo dos o tres grupos de prestaciones relacionadas.
- Scrum: Cada iteración tiende a ser relativamente corta (por ejemplo, horas, días o unas pocas semanas), y los incrementos de las prestaciones son proporcionalmente pequeños, como unas pocas mejoras y/o dos o tres prestaciones nuevas.
- Kanban: Implementado con o sin iteraciones de longitud fija, que puede ofrecer una sola mejora o prestación una vez finalizada, o puede agrupar prestaciones para liberarlas de una sola vez.
- Espiral (o prototipado): Implica la creación de incrementos experimentales, algunos de los cuales pueden ser reelaborados en profundidad o incluso abandonados en trabajos de desarrollo posteriores.

Los componentes o sistemas desarrollados utilizando estos métodos, a menudo, implican niveles de prueba que se solapan e iteran a lo largo de todo el desarrollo. Lo ideal es que cada prestación se pruebe en varios niveles de prueba a medida que se aproxima la entrega. En algunos casos, los equipos utilizan la entrega continua o el despliegue continuo, lo que implica una automatización significativa de múltiples niveles de prueba como parte de sus canales de entrega[18]. Muchos de los esfuerzos de desarrollo que utilizan estos métodos también incluyen el concepto de equipos auto-organizados, 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 generan un sistema en crecimiento, que puede ser liberado a los usuarios finales prestación a prestación, iteración a iteración, o en la forma de un lanzamiento a gran escala más tradicional. Independientemente de si los incrementos de software se liberan a los usuarios finales, la prueba de regresión es cada vez más importante a medida que el sistema crece.

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

Para más información sobre las pruebas de software en el contexto del desarrollo ágil, véase Programa de Estudio de Probador Certificado del ISTQB® de Nivel Básico, Extensión Ágil (ISTQB-AT por sus siglas en inglés) del ISTQB, Black 2017, Crispin 2008, y Gregory 2015.

2.1.2 Modelos de Ciclo de Vida del 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 apropiado del ciclo de vida del desarrollo de software en función del objetivo del proyecto, el tipo de producto que se está desarrollando, las prioridades del negocio (por ejemplo, el tiempo de comercialización), y los riesgos de producto y de proyecto identificados. Por ejemplo, el desarrollo y la prueba de un sistema de gestión interna menor debe diferir del desarrollo y la prueba de un sistema crítico para la seguridad física, como el sistema de control de frenos de un automóvil. Como otro ejemplo, en algunos casos las cuestiones[19] relativas a la organización y de índole cultural 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 comercial de distribución masiva (COTS por sus siglas en inglés) en un sistema más amplio, el comprador puede realizar pruebas de interoperabilidad a nivel de prueba de integración de sistemas (por ejemplo, integración en la infraestructura y otros sistemas) y a nivel de prueba de aceptación (funcional y no funcional, junto con la prueba de aceptación del usuario y la prueba de aceptación operativa). Véase la sección 2.2 para una descripción de los niveles de prueba y la sección 2.3 para una descripción de los tipos de prueba.

Además, se pueden combinar los propios modelos de ciclo de vida de desarrollo de software. Por ejemplo, un modelo en V puede ser usado para el desarrollo y prueba de los sistemas backend y sus integraciones, mientras que un modelo de desarrollo ágil puede ser usado para desarrollar y probar la interfaz de usuario (UI) y funcionalidad del front-end. El prototipado puede ser utilizado en las primeras etapas de un proyecto, adoptando un modelo de desarrollo incremental una vez finalizada la fase experimental.

[18] **Consultar ****
[19] **Consultar ****

Los sistemas de Internet de las Cosas[20] (IoT por sus siglas en inglés), que consisten en muchos objetos diferentes, tales como dispositivos, productos y servicios, suelen aplicar modelos de ciclo de vida de desarrollo de software separados para cada objeto. Esto supone un reto especial para el desarrollo de las versiones de sistemas de Internet de las Cosas. Además, el ciclo de vida de desarrollo de software de dichos objetos pone un mayor énfasis en las fases posteriores del ciclo de vida de desarrollo de software después de que se hayan introducido para su uso operativo (por ejemplo, las fases de operación, actualización y retirada).

2.2 Niveles de Prueba

Los niveles de prueba son grupos de actividades de prueba que se organizan y gestionan conjuntamente. 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 determinado, desde unidades o componentes individuales hasta sistemas completos o, en su caso, sistemas de sistemas. Los niveles de prueba están relacionados con otras actividades dentro del ciclo de vida de desarrollo de software. Los niveles de prueba utilizados en este programa de estudio son:

- Prueba de componente.
- Prueba de integración.
- Prueba de sistema.
- Prueba de aceptación.

Los niveles de prueba se caracterizan por los siguientes atributos:

- Objetivos específicos.
- Bases de prueba, referenciadas para generar casos de prueba.
- Objeto de prueba (es decir, lo que se está probando).
- Defectos y fallos característicos.
- Enfoques y responsabilidades específicos.

Se requiere un entorno de prueba adecuado para cada nivel de prueba. En la prueba de aceptación, por ejemplo, un entorno de prueba similar al de producción es ideal, mientras que en la prueba de componente los desarrolladores suelen utilizar su propio entorno de desarrollo.

[20] **Consultar ****

2.2.1 Prueba de Componente

Objetivos de la Prueba de Componente

La prueba de componente (también conocida como prueba unitaria o de módulo) se centra en los componentes que se pueden probar por separado. Los objetivos de la prueba de componente incluyen:

- Reducir el riesgo.
- Verificar que los comportamientos funcionales y no funcionales del componente son los diseñados y especificados.
- Generar confianza en la calidad del componente.
- Encontrar defectos en el componente.
- Prevenir la propagación de defectos a niveles de prueba superiores.

En algunos casos, especialmente en modelos de desarrollo incrementales e iterativos (por ejemplo, Ágiles) donde los cambios de código son continuos, la prueba de regresión de componente automatizada juega un papel clave en la construcción de la confianza en que los cambios no han dañado a los componentes existentes.

La prueba de componente, a menudo, se realiza de forma aislada del resto del sistema, dependiendo del modelo de ciclo de vida de desarrollo de software y del sistema, lo que puede requerir objetos simulados, virtualización de servicios, arneses, stubs y controladores. La prueba de componente pueden cubrir la funcionalidad (por ejemplo, la exactitud de los cálculos), las características no funcionales (por ejemplo, la búsqueda de fugas de memoria) y las propiedades estructurales (por ejemplo, pruebas de decisión).

Bases de Prueba

Algunos ejemplos de productos de trabajo que se pueden utilizar como base de prueba para la prueba de componente incluyen:

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

Objetos de Prueba

Los objetos de prueba característicos para la prueba de componente incluyen:

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

Defectos y Fallos Característicos

Ejemplos de defectos y fallos característicos de la prueba de componente incluyen:

- Funcionamiento incorrecto (por ejemplo, no como se describe en las especificaciones de diseño).
- Problemas de flujo de datos.
- Código y lógica incorrectos.

Por lo general, los defectos se corrigen 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 Específicos

En general, el desarrollador que escribió el código realiza la prueba de componente, pero al menos requiere acceso al código que se está probando. Los desarrolladores pueden alternar el desarrollo de componentes con la búsqueda y corrección de defectos. A menudo, los desarrolladores escriben y ejecutan pruebas después de haber escrito el código de un componente. Sin embargo, especialmente en el desarrollo Ágil, la redacción de casos de prueba de componente automatizados puede preceder a la redacción del código de la aplicación.

Por ejemplo, considerar el desarrollo guiado por pruebas (TDD por sus siglas en inglés). El desarrollo guiado por pruebas es altamente iterativo y se basa en ciclos de desarrollo de casos de prueba automatizados, luego se construyen e integran pequeños fragmentos de código, a continuación, se ejecuta la prueba de componente, se corrige cualquier cuestión y se refactoriza el código. Este proceso continúa hasta que el componente ha sido completamente construido y ha pasado toda la prueba de componente. El desarrollo guiado por pruebas es un ejemplo de un enfoque del tipo "prueba primero". Aunque el desarrollo guiado por pruebas se originó en eXtreme Programming (XP), se ha extendido a otras formas Agiles y también a ciclos de vida secuenciales (ver el Programa de Estudio de Probador Certificado del ISTQB® de Nivel Básico, Extensión Ágil - ISTQB-AT).

2.2.2 Prueba de Integración

Objetivos de la Prueba de Integración

La prueba de integración se centra en las interacciones entre componentes o sistemas. Los objetivos de la prueba de integración incluyen:

- Reducir el riesgo.
- Verificar que los comportamientos funcionales y no funcionales de las interfaces sean los diseñados y especificados.
- Generar confianza en la calidad de las interfaces.
- Encontrar defectos (que pueden estar en las propias interfaces o dentro de los componentes o sistemas).
- Prevenir la propagación de defectos a niveles de prueba superiores.

De la misma forma que con la prueba de componente, en algunos casos la prueba de regresión de integración automatizada proporciona la confianza de que los cambios no han dañado a las interfaces, los componentes o los sistemas existentes.

Hay dos niveles de prueba de integración diferentes que se describen en este programa de estudio y que se pueden llevar a cabo sobre objetos de prueba de diferentes tamaños de la siguiente manera:

- La prueba de integración de componentes se centran en las interacciones e interfaces entre los componentes integrados. La prueba de integración de componentes se realiza después de las prueba de componente y, en general, está automatizada. En el desarrollo iterativo e incremental, la prueba de integración de componentes suele formar parte del proceso de integración continua.
- La prueba de integración de sistemas se centran en las interacciones e interfaces entre sistemas, paquetes y microservicios. La prueba de integración de sistemas también pueden cubrir las interacciones con, e interfaces proporcionadas por, organizaciones externas (por ejemplo, servicios web). En este caso, la organización que desarrolla no controla las interfaces externas, lo que puede crear varios obstáculos para la prueba (por ejemplo, asegurar que se resuelvan los defectos de bloqueo de pruebas en el código de la organización externa, organizar los entornos de pruebas, etc.). La prueba de integración de sistemas pueden realizarse después de la prueba de sistema o en paralelo con las actividades de prueba de sistema en curso (tanto en el desarrollo secuencial como en el desarrollo iterativo e incremental).

Bases de Prueba

Algunos ejemplos de productos de trabajo que pueden utilizarse como base de prueba para la prueba de integración incluyen:

- Diseño de software y sistemas.
- Diagramas de secuencia.
- Especificaciones de interfaz y protocolos de comunicación.
- Casos de uso.
- Arquitectura a nivel de componente o de sistema.
- Flujos de trabajo.
- Definiciones de interfaces externas.

Objetos de Prueba

Los objetos de prueba característicos para la prueba de integración incluyen:

- Subsistemas.
- Bases de datos.
- Infraestructura.
- Interfaces.
- Interfaces de programación de aplicaciones (API por sus siglas en inglés).
- Microservicios.

Defectos y Fallos Característicos

Entre los ejemplos de defectos y fallos característicos de la prueba de integración de componentes se incluyen los siguientes:

- Datos incorrectos, datos faltantes o codificación incorrecta de datos.
- Secuenciación o sincronización incorrecta de las llamadas a la interfaz.
- Incompatibilidad de la interfaz.
- Fallos en la comunicación entre componentes.
- Fallos de comunicación entre componentes no tratados o tratados de forma incorrecta.
- Suposiciones incorrectas sobre el significado, las unidades o las fronteras de los datos que se transmiten entre componentes.

Entre los ejemplos de defectos y fallos característicos de la prueba de integración de sistemas se incluyen los siguientes:

- Estructuras de mensajes inconsistentes entre sistemas.
- Datos incorrectos, datos faltantes o codificación incorrecta de datos.
- Incompatibilidad de la interfaz.
- Fallos en la comunicación entre sistemas.
- Fallos de comunicación entre sistemas no tratados o tratados de forma incorrecta. Suposiciones incorrectas sobre el significado, las unidades o las fronteras de los datos que se transmiten entre sistemas.
- Incumplimiento de las normas de seguridad obligatorias.

Enfoques y Responsabilidades Específicos

La prueba de integración de componentes y la prueba de integración de sistemas deben concentrarse en la integración propiamente dicha. Por ejemplo, si se integra el módulo A con el módulo B, la prueba debe centrarse en la comunicación entre los módulos, no en la funcionalidad de los módulos individuales, como debería haberse hecho durante la prueba de componente. Si se integra el sistema X con el sistema Y, la prueba debe centrarse en la comunicación entre los sistemas, no en la funcionalidad de los sistemas individuales, como debería haberse hecho durante la prueba de sistema. Se puede utilizar los tipos de prueba funcional, no funcional y estructural.

La prueba de integración de componentes suele ser responsabilidad de los desarrolladores. La prueba de integración de sistemas es, en general, responsabilidad de los probadores. En condiciones ideales, los probadores que realizan la prueba de integración de sistemas deberían entender la arquitectura del sistema y deberían haber influido en la planificación de la integración.

Si la prueba de integración y la estrategia de integración se planifican antes de que se construyan los componentes o sistemas, estos componentes o sistemas se pueden construir en el orden adecuado para que la prueba sea más eficiente. Las estrategias de integración sistemática pueden basarse en la arquitectura del sistema (por ejemplo, ascendente[21] y descendente[22]), en tareas funcionales, en secuencias de procesamiento de transacciones o en algún otro aspecto del sistema o de los componentes. Para simplificar el aislamiento de defectos y detectar defectos de forma temprana, la integración debe ser normalmente incremental (es decir, un pequeño número de componentes o sistemas adicionales a la vez) en lugar de "big bang" (es decir, la integración de todos los componentes o sistemas en un solo paso). Un análisis de riesgo de las interfaces más complejas puede ayudar a centrar la prueba de integración.

---

[21] **Consultar ****

[22] **Consultar ****

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

2.2.3 Prueba de Sistema

Objetivos de la Prueba de Sistema

La prueba de sistema se centra en el comportamiento y las capacidades de todo un sistema o producto, a menudo teniendo en cuenta las tareas extremo a extremo que el sistema puede realizar y los comportamientos no funcionales que exhibe mientras realiza esas tareas. Los objetivos de la prueba de sistema incluyen:

- Reducir el riesgo.
- Verificar que los comportamientos funcionales y no funcionales del sistema son los diseñados y especificados.
- Validar que el sistema está completo y que funcionará como se espera.
- Generar confianza en la calidad del sistema considerado como un todo.
- Encontrar defectos.
- Prevenir la propagación de defectos a niveles de prueba superiores o a producción.

Para ciertos sistemas, la verificación de la calidad de los datos puede ser un objetivo. Al igual que con la prueba de componente y la prueba de integración, en algunos casos la prueba de regresión de sistema automatizada proporciona la confianza de que los cambios no han dañado características existentes o capacidades extremo a extremo. A menudo, la prueba de sistema produce información que es utilizada por los implicados para tomar decisiones con respecto al lanzamiento. La prueba de sistema también puede satisfacer requisitos o estándares legales o regulatorios.

El entorno de prueba debe corresponder, en condiciones ideales, al entorno objetivo final o entorno de producción.

Bases de Prueba

Algunos ejemplos de productos de trabajo que se pueden utilizar como base de prueba para la prueba de sistema incluyen:

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

Objetos de Prueba

Los objetos de prueba característicos para la prueba de sistema incluyen:

- Aplicaciones.
- Sistemas hardware/software.
- Sistemas operativos.
- Sistema sujeto a prueba (SSP).
- Configuración del sistema y datos de configuración.

Defectos y Fallos Característicos

Entre los ejemplos de defectos y fallos característicos de la prueba de sistema se incluyen los siguientes:

- Cálculos incorrectos.
- Comportamiento funcional o no funcional del sistema incorrecto o inesperado.
- Control y/o flujos de datos incorrectos dentro del sistema.
- Incapacidad para llevar a cabo, de forma adecuada y completa, las tareas funcionales extremo a extremo.
- Fallo del sistema para operar correctamente en el/los entorno/s de producción.
- Fallo del sistema para funcionar como se describe en los manuales del sistema y de usuario.

Enfoques y Responsabilidades Específicos

La prueba de sistema debe centrarse en el comportamiento global y extremo a extremo del sistema en su conjunto, tanto funcional como no funcional. La prueba de sistema debe utilizar las técnicas más apropiadas (véase el capítulo 4) para los aspectos del sistema que serán probados. Por ejemplo, se puede crear una tabla de decisión para verificar si el comportamiento funcional es el descrito en términos de reglas de negocio.

Los probadores independientes, en general, llevan a cabo la prueba de sistema. Los defectos en las especificaciones (por ejemplo, la falta de historias de usuario, los requisitos de negocio establecidos de forma incorrecta, etc.) pueden llevar a una falta de comprensión o a desacuerdos sobre el comportamiento esperado del sistema. Tales situaciones pueden causar falsos positivos y falsos negativos, lo que hace perder tiempo y reduce la eficacia de la detección de defectos, respectivamente. La participación temprana de los probadores en el perfeccionamiento de la historia de usuario o en actividades de prueba estática, como las revisiones, ayuda a reducir la incidencia de tales situaciones.

2.2.4 Prueba de Aceptación

Objetivos de la Prueba de Aceptación

La prueba de aceptación, al igual que la prueba de sistema, se centra normalmente en el comportamiento y las capacidades de todo un sistema o producto. Los objetivos de la prueba de aceptación incluyen:

- Establecer confianza en la calidad del sistema en su conjunto.
- Validar que el sistema está completo y que funcionará como se espera.
- Verificar que los comportamientos funcionales y no funcionales del sistema sean los especificados.

La prueba de aceptación puede producir información para evaluar el grado de preparación del sistema para su despliegue y uso por parte del cliente (usuario final). Los defectos pueden encontrarse durante las prueba de aceptación, pero encontrar defectos no suele ser un objetivo, y encontrar un número significativo de defectos durante la prueba de aceptación puede, en algunos casos, considerarse un riesgo importante para el proyecto. La prueba de aceptación también pueden satisfacer requisitos o normas legales o reglamentarios.

Las formas comunes de prueba de aceptación incluyen las siguientes:

- Prueba de aceptación de usuario.
- Prueba de aceptación operativa.
- Prueba de aceptación contractual y de regulación.
- Prueba alfa y beta.

Cada uno de estas se describe en las siguientes cuatro subsecciones.

Prueba de Aceptación de Usuario (UAT por sus siglas en inglés)

La prueba de aceptación de sistema por parte de los usuarios se centra normalmente en la validación de la idoneidad 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 utilizar el sistema para satisfacer sus necesidades, cumplir con los requisitos y realizar los procesos de negocio con la mínima dificultad, coste y riesgo.

Prueba de Aceptación Operativa

La prueba de aceptación de sistema por parte del personal de operaciones o de la administración del sistema se realiza, por lo general, en un entorno de producción (simulado). La prueba se centra en los aspectos operativos, y pueden incluir:

- Prueba de copia de seguridad y restauración.
- Instalación, desinstalación y actualización.
- Recuperación ante desastres.
- Gestión de usuarios.
- Tareas de mantenimiento.
- Carga de datos y tareas de migración.
- Comprobación de vulnerabilidades de seguridad.
- Prueba de rendimiento.

El objetivo principal de la prueba 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.

Prueba de Aceptación Contractual y Normativa

La prueba de aceptación contractual se realiza en función de los criterios de aceptación del contrato para el desarrollo de software a medida. Los criterios de aceptación deben definirse cuando las partes acuerdan el contrato. La prueba de aceptación contractual suele ser realizada por usuarios o por probadores independientes.

La prueba de aceptación normativa se lleva a cabo con respecto a cualquier norma que deba cumplirse, como las normas gubernamentales, legales o de seguridad física. La prueba de aceptación normativa suele ser realizada por los usuarios o por probadores independientes, en ocasiones los resultados son presenciados o auditados por agencias reguladoras.

El principal objetivo de la prueba de aceptación contractual y normativa es crear confianza en que se ha logrado la conformidad contractual o normativa.

Pruebas Alfa y Beta

Las pruebas alfa y beta suelen ser utilizadas por los desarrolladores de software comercial de distribución masiva (COTS por sus siglas en inglés) que desean obtener retroalimentación de los usuarios, clientes y/u operadores potenciales o existentes antes de que el producto de software sea puesto en el mercado. La prueba alfa se realiza en las instalaciones de la organización que desarrolla, no por el equipo de desarrollo, sino por clientes potenciales o existentes, y/u operadores o un equipo de prueba independiente. La prueba beta es realizada por clientes potenciales o existentes, y/u operadores en sus propias instalaciones. La prueba beta puede tener lugar después de la prueba alfa, o puede ocurrir sin que se haya realizado ninguna prueba alfa previa.

Uno de los objetivos de las pruebas alfa y beta es generar confianza entre los clientes potenciales o existentes y/u operadores de que pueden utilizar el sistema en condiciones normales y cotidianas, así como en el entorno o los entornos operativos, para lograr sus objetivos con la mínima dificultad, coste y riesgo. Otro objetivo puede ser la detección de defectos relacionados con las condiciones y el entorno o los entornos en los que se utilizará el sistema, especialmente cuando las condiciones y los entornos sean difíciles de reproducir por parte del equipo de desarrollo.

Base de Prueba

Entre los ejemplos de productos de trabajo que se pueden utilizar como base de prueba para cualquier forma de prueba de aceptación se encuentran:

- Procesos de negocio.
- Requisitos de usuario o de negocio.
- Normativas, contratos legales y estándares.
- Casos de uso.
- Requisitos de sistema.
- Documentación del sistema o del usuario.
- Procedimientos de instalación.
- Informes de análisis de riesgo.

Además, como base de prueba para derivar casos de prueba para la prueba de aceptación operativa, se pueden utilizar 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 operaciones.
- Instrucciones de despliegue e instalación.
- Objetivos de rendimiento.
- Paquetes de base de datos.
- Estándares o reglamentos de seguridad.

Objetos de Prueba Característicos

Los objetos de prueba característicos para cualquier forma de prueba 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 críticos (para pruebas de continuidad del negocio y recuperación de desastres).
- Procesos operativos y de mantenimiento.
- Formularios.
- Informes.
- Datos de producción existentes y transformados.

Defectos y Fallos Característicos

Entre los ejemplos de defectos característicos de cualquier forma de prueba de aceptación se encuentran:

- Los flujos de trabajo del sistema no cumplen con los requisitos de negocio o de usuario.
- Las reglas de negocio no se implementan de forma correcta.
- El sistema no satisface los requisitos contractuales o reglamentarios.
- Fallos no funcionales tales como vulnerabilidades de seguridad, eficiencia de rendimiento inadecuada bajo cargas elevadas o funcionamiento inadecuado en una plataforma soportada.

Enfoques y Responsabilidades Específicos

La prueba de aceptación es, a menudo, responsabilidad de los clientes, usuarios de negocio, propietarios de producto u operadores de un sistema, y otros implicados también pueden estar involucrados.

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

- La prueba de aceptación de un producto software comercial de distribución masiva (COTS por sus siglas en inglés) puede tener lugar cuando se instala o integra.
- La prueba de aceptación de una mejora funcional nueva puede tener lugar antes de la prueba de sistema.

En el desarrollo iterativo, los equipos de proyecto pueden emplear varias formas de prueba de aceptación durante y al final de cada iteración, como las que se centran en verificar una nueva característica en relación con sus criterios de aceptación y las que se centran en validar que una nueva característica satisface las necesidades de los usuarios. Además, se pueden realizar pruebas alfa y beta, ya sea al final de cada iteración, después de la finalización de cada iteración, o después de una serie de iteraciones. La prueba de aceptación de usuario, prueba de aceptación operativa, prueba de aceptación normativa y prueba de aceptación contractual también pueden t, ya sea al cierre de cada iteración, después de la finalización de 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 de una parte de un sistema, basadas en objetivos de prueba específicos. Dichos objetivos pueden incluir:

- Evaluar las características de calidad funcional, como la completitud, corrección y pertinencia.
- Evaluar características no funcionales de calidad, tales como fiabilidad, eficiencia de desempeño, 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, tales como confirmar que los defectos han sido corregidos (prueba de confirmación) y buscar cambios no deseados en el comportamiento que resulten de cambios en el software o en el entorno (prueba de regresión).

2.3.1 Prueba Funcional

La prueba funcional de un sistema incluye pruebas que evalúan las funciones que el sistema debe realizar. Los requisitos funcionales pueden estar descritos en productos de trabajo tales como especificaciones de requisitos de negocio, épicas, historias de usuario, casos de uso, o especificaciones funcionales, o pueden estar sin documentar. Las funciones describen "qué" debe hacer el sistema.

Se deben realizar pruebas funcionales en todos los niveles de prueba (por ejemplo, la prueba de componente puede basarse en una especificación de componentes), aunque el enfoque es diferente en cada nivel (véase la sección 2.2).

La prueba funcional observa el comportamiento del software, por lo que se pueden utilizar técnicas de caja negra para obtener las condiciones de prueba y los casos de prueba para la funcionalidad del componente o sistema (véase la sección 4.2).

Se puede medir la intensidad de la prueba funcional a través de la cobertura funcional. La cobertura funcional es la medida en que algún tipo de elemento funcional ha sido practicado por pruebas, y se expresa como un porcentaje del tipo o tipos de elemento cubiertos. Por ejemplo, utilizando la trazabilidad entre pruebas y requisitos funcionales, se puede calcular el porcentaje de estos requisitos abordados por las pruebas, identificando potencialmente carencias en la cobertura.

El diseño y la ejecución de pruebas funcionales pueden implicar competencias o conocimientos especiales, como el conocimiento del problema de negocio específico que resuelve el software (por ejemplo, el software de modelado geológico para las industrias del petróleo y el gas) o el papel particular que desempeña el software (por ejemplo, el software de juegos de azar de ordenador que proporciona entretenimiento interactivo).

2.3.2 Prueba No Funcional

La prueba no funcional de un sistema evalúa las características de sistemas y software, como la usabilidad, la eficiencia del desempeño o la seguridad. Consulte el estándar ISO (ISO/IEC 25010) para una clasificación de las características de calidad de producto software. La prueba no funcional prueba "cómo de bien" se comporta el sistema[23].

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

Se pueden utilizar técnicas de caja negra (véase la sección 4.2) para obtener condiciones de prueba y casos de prueba para pruebas no funcionales. Por ejemplo, se puede utilizar el análisis de valores frontera para definir las condiciones de estrés para las pruebas de rendimiento.

Se puede medir la intensidad de la prueba no funcional a través de la cobertura no funcional. La cobertura no funcional es la medida en que algún tipo de elemento no funcional ha sido practicado por pruebas, y se expresa como un porcentaje del tipo o tipos de elemento cubiertos. Por ejemplo, utilizando la trazabilidad entre las pruebas y los dispositivos compatibles con una aplicación móvil, se puede calcular el porcentaje de dispositivos tratados por las pruebas de compatibilidad, identificando potencialmente las carencias en la cobertura.

El diseño y la ejecución de la prueba no funcional pueden implicar competencias o conocimientos especiales, como el conocimiento de las debilidades inherentes a un diseño o tecnología (por ejemplo, vulnerabilidades de seguridad asociadas con determinados lenguajes de programación) o la base de usuarios concreta (por ejemplo, la persona[24] de los usuarios de los sistemas de gestión de centros sanitarios).

Consultar los programas de estudios de Probador Certificado de Nivel Avanzado - Analista de Pruebas de ISTQB, Probador Certificado de Nivel Avanzado - Analista de Pruebas Técnicas de ISTQB, Probador Certificado de Nivel Avanzado - Probador de la Seguridad de ISTQB y otros módulos especializados de ISTQB para obtener más detalles sobre las pruebas de características de la calidad no funcionales.

[23] **Consultar ****
[24] **Consultar ****

2.3.3 Prueba de Caja Blanca

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

Se puede medir la intensidad de la prueba de caja blanca a través de la cobertura estructural. La cobertura estructural es la medida en que algún tipo de elemento estructural ha sido practicado mediante pruebas, y se expresa como un porcentaje del tipo de elemento cubierto.

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

El diseño y la ejecución de la prueba de caja blanca pueden implicar competencias o conocimientos especiales, como la forma en que se construye el código (por ejemplo, para utilizar herramientas de cobertura de código), cómo se almacenan los datos (por ejemplo, para evaluar posibles consultas a la base de datos), y cómo utilizar las herramientas de cobertura e interpretar correctamente sus resultados.

2.3.4 Prueba Asociada al Cambio

Cuando se realizan cambios en un sistema, ya sea para corregir un defecto o debido a una funcionalidad nueva o modificada, se debe probar para confirmar que los cambios han corregido el defecto o implementado la funcionalidad correctamente, y no han causado ninguna consecuencia adversa imprevista.

- Prueba de confirmación: Una vez corregido un defecto, el software se puede probar con todos los casos de prueba que fallaron debido al defecto, que se deben volver a ejecutar en la nueva versión de software. El software también puede probarse con nuevas pruebas si, por ejemplo, el defecto consistía en la falta de una funcionalidad. Como mínimo, los pasos para reproducir el fallo o los fallos causados por el defecto deben volver a ejecutarse en la nueva versión del software. El objetivo de una prueba de confirmación es confirmar que el defecto original se ha solucionado de forma satisfactoria.
- Prueba de regresión: Es posible que un cambio hecho en una parte del código, ya sea una corrección 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 modificaciones en el entorno, tales como una nueva versión de un sistema operativo o de un sistema de gestión de bases de datos. Estos efectos secundarios no deseados se denominan regresiones. La prueba de regresión implica la realización de pruebas para detectar estos efectos secundarios no deseados.

La prueba de confirmación y la prueba de regresión se realizan en todos los niveles de prueba.

Especialmente en los ciclos de vida de desarrollo iterativos e incrementales (por ejemplo, Agile), las nuevas características, los cambios en las características existentes y la refactorización del código dan como resultado cambios frecuentes en el código, lo que también requiere pruebas asociadas al cambio. Debido a la naturaleza evolutiva del sistema, la prueba de confirmación y la prueba regresión son muy importantes. Esto es particularmente relevante para los sistemas de Internet de las Cosas, donde los objetos individuales (por ejemplo, los dispositivos) se actualizan o reemplazan con frecuencia.

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

2.3.5 Tipos de Prueba y Niveles de Prueba

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

- Para la prueba de componente, las pruebas se diseñan en base a la forma en que un componente debe calcular el interés compuesto.
- Para la prueba de integración de componentes, las pruebas se diseñan en función de cómo la información de la cuenta capturada en la interfaz de usuario se transfiere a la lógica de negocio.
- Para la prueba de sistema, las pruebas se diseñan en base a cómo los titulares de cuentas pueden solicitar una línea de crédito sobre sus cuentas corrientes.
- Para la prueba de integración de sistemas, las pruebas se diseñan en función de cómo el sistema utiliza un microservicio externo para comprobar la calificación crediticia del titular de una cuenta.
- Para la prueba de aceptación, las pruebas se diseñan en base a la forma en que el empleado del banco tramita la aprobación o rechazo de una solicitud de crédito.

Los siguientes son ejemplos de pruebas no funcionales:

- Para la prueba de componente, las pruebas de rendimiento están diseñadas para evaluar el número de ciclos de CPU necesarios para realizar un cálculo de intereses totales complejo.
- Para la prueba de integración de componentes, las pruebas de seguridad están diseñadas para vulnerabilidades de desbordamiento de memoria intermedia debido a que los datos pasan de la interfaz de usuario a la lógica de negocio.
- Para la prueba de sistema, las pruebas de portabilidad están diseñadas para comprobar si la capa de presentación funciona en todos los navegadores y dispositivos móviles soportados.
- Para la prueba de integración de sistemas, las pruebas de fiabilidad están diseñadas para evaluar la solidez del sistema en caso de que el microservicio de calificación crediticia falle en responder.
- Para la prueba de aceptación, las pruebas de usabilidad están diseñadas para evaluar la accesibilidad de la interfaz de procesamiento de crédito bancario para personas con discapacidades.

Los siguientes son ejemplos de prueba de caja blanca:

- Para la prueba de componente, las pruebas están diseñadas para lograr una cobertura completa de sentencia y decisión (ver sección 4.3) para todos los componentes que realizan cálculos financieros.
- Para la prueba de integración de componentes, las pruebas están diseñadas para practicar cómo cada pantalla de la interfaz del navegador pasa datos a la siguiente pantalla y a la lógica de negocio.
- Para la prueba de sistema, las pruebas están diseñadas para cubrir las secuencias de páginas web que pueden ocurrir durante una solicitud de línea de crédito.

- Para la prueba de integración de sistemas, las pruebas están diseñadas para practicar todos los tipos de consulta posibles que se envían al microservicio de calificación crediticia.
- Para la prueba de aceptación, las pruebas están diseñadas para cubrir todas las estructuras de archivos de datos financieros soportados y rangos de valores para transferencias de banco-a-banco.

Por último, los siguientes son ejemplos de pruebas asociadas al cambio:

- Para la prueba de componente, se construyen pruebas de regresión automatizadas para cada componente y se incluyen dentro del marco de integración continua.
- Para la prueba de integración de componentes, las pruebas están diseñadas para confirmar la corrección de defectos relacionados con la interfaz a medida que las correcciones se registran en el repositorio de código.
- Para la prueba de sistema, todas las pruebas de un flujo de trabajo dado se ejecutan de nuevo si cambia alguna pantalla de ese flujo de trabajo.
- Para la prueba de integración de sistemas, las pruebas de la aplicación que interactúa con el microservicio de calificación de crédito se vuelven a ejecutar diariamente como parte del despliegue continuo de ese microservicio.
- Para la prueba de aceptación, todas las pruebas que han fallado previamente se vuelven a ejecutar después de que se haya corregido un defecto encontrado en la prueba de aceptación.

Aunque esta sección proporciona ejemplos de cada tipo de prueba en todos los niveles, no es necesario, para todo software, que cada tipo de prueba esté presente en todos los niveles. Sin embargo, es importante ejecutar los tipos de prueba aplicables en cada nivel, especialmente en el nivel más temprano en el que tiene lugar el tipo de prueba.

2.4 Prueba de Mantenimiento

Una vez desplegados en los entornos de producción, es necesario mantener el software y los sistemas. Los cambios de diversa índole son casi inevitables en el software y en los sistemas entregados, ya sea para corregir defectos descubiertos en el uso operativo, para añadir nuevas funcionalidades o para eliminar o alterar funcionalidades ya entregadas. El mantenimiento también es necesario para preservar o mejorar las características de calidad no funcionales del componente o sistema a lo largo de su vida útil, especialmente en lo que se refiere a eficiencia de desempeño, compatibilidad, fiabilidad, seguridad, compatibilidad y portabilidad.

Cuando se realizan cambios como parte del mantenimiento, se debe realizar una prueba de mantenimiento, tanto para evaluar el éxito con el que se realizaron los cambios como para comprobar los posibles efectos secundarios (por ejemplo, regresiones) en las partes del sistema que permanecen inalteradas (que suele ser la mayor parte del sistema). La prueba de mantenimiento se centra en probar los cambios en el sistema, así como en probar las piezas no modificadas que podrían haberse visto afectadas por los cambios. El mantenimiento puede incluir lanzamientos planificados y no planificados (soluciones en caliente[25]).

[25] **Consultar ****

El lanzamiento de un mantenimiento puede requerir probar el mantenimiento en múltiples niveles de prueba, utilizando varios tipos de prueba, según su alcance. El alcance de la prueba de mantenimiento depende de:

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

2.4.1 Activadores para el Mantenimiento

Existen varias razones por las que el mantenimiento del software y, por lo tanto, la prueba de mantenimiento, se llevan a cabo, tanto para las modificaciones planificadas como para las no planificadas.

Se puede clasificar los activadores de mantenimiento de la siguiente manera:

- Modificación, tales como mejoras planificadas (por ejemplo, basadas en lanzamientos), cambios correctivos y de emergencia, cambios en el entorno operativo (tales como actualizaciones previstas del sistema operativo o de la base de datos), actualizaciones del software comercial de distribución masiva ("COTS" por sus siglas en inglés), y parches para los defectos y las vulnerabilidades.
- Migración, por ejemplo, de una plataforma a otra, que puede requerir pruebas operativas del nuevo entorno y del software modificado, o pruebas de conversión de datos cuando los datos de otra aplicación se migren al sistema en mantenimiento.
- Retirada, por ejemplo, cuando una aplicación llega al final de su vida útil.

Cuando se retira una aplicación o sistema, esto puede requerir la comprobación de la migración de datos o el archivo si se requieren largos períodos de retención de datos[26]. También puede ser necesario probar los procedimientos de restauración/recuperación después de archivar durante largos períodos de retención. Además, es posible que sea necesario realizar una prueba de regresión para garantizar que cualquier funcionalidad que permanezca en servicio siga funcionando.

En el caso de los sistemas de Internet de las Cosas, la prueba de mantenimiento puede activarse por la introducción de cosas completamente nuevas o modificadas, tales como dispositivos de hardware y servicios de software, en el sistema global. La prueba de mantenimiento de estos sistemas hace especial énfasis en la prueba de integración a diferentes niveles (por ejemplo, nivel de red, nivel de aplicación) y en los aspectos de seguridad, en particular los relativos a los datos personales.

2.4.2 Análisis de Impacto para el Mantenimiento

El análisis de impacto evalúa los cambios que se hicieron para el lanzamiento de un mantenimiento con el objetivo de identificar las consecuencias previstas, así como los efectos secundarios esperados y posibles de un cambio, y para identificar las áreas del 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 necesitan ser probados para las regresiones, posiblemente después de haber actualizado cualquier prueba existente afectada por el cambio.

[26] **Consultar ****

Se puede realizar un análisis de impacto antes de realizar un cambio, para ayudar a decidir si se debe realizar el cambio, basándose en las consecuencias potenciales en otras áreas del sistema.

El análisis de impacto puede ser difícil si:

- Las especificaciones (por ejemplo, requisitos de negocio, historias de usuario, arquitectura) están desactualizadas o no existen.
- Los casos de prueba no están documentados o están desactualizados.
- No se ha mantenido la trazabilidad bidireccional entre las pruebas y la base de prueba.
- El soporte de herramientas es débil o inexistente.
- Las personas involucradas no tienen conocimientos de dominio y/o sistema.
- No se ha prestado suficiente atención a la mantenibilidad del software durante el desarrollo.

3 Prueba Estática

Duración: 135 minutos

Palabras Clave

inspección ("inspection")
lectura basada en perspectiva ("perspective-based reading")
prueba dinámica ("dynamic testing")
prueba estática ("static testing")
revisión ("review")
revisión ad hoc ("ad hoc reviewing")
revisión basada en escenarios ("scenario-based reviewing")
revisión basada en lista de comprobación ("checklist-based reviewing")
revisión basada en roles ("role-based reviewing")
revisión estática ("static analysis")
revisión formal ("formal review")
revisión guiada ("walkthrough")
revisión informal ("informal review")
revisión técnica ("technical review")

Objetivos de Aprendizaje para la Prueba Estática

3.1 Prueba Estática: Conceptos Básicos

NB-3.1.1 (K1) Reconocer los tipos de productos de trabajo de software que pueden ser examinados por las diferentes técnicas de prueba estática.

NB-3.1.2 (K2) Utilizar ejemplos para describir el valor de la prueba estática.

NB-3.1.3 (K2) Explicar la diferencia entre técnicas estáticas y dinámicas, considerando los objetivos, los tipos de defectos a identificar y el papel de estas técnicas dentro del ciclo de vida del software.

3.2 Proceso de Revisión

NB-3.2.1 (K2) Resumir las actividades del proceso de revisión de productos de trabajo.

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

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

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

NB-3.2.5 (K2) Explicar los factores que contribuyen al éxito de la revisión.

3.1 Prueba Estática: Conceptos Básicos

A diferencia de la prueba dinámica, que requiere la ejecución del software que se está probando, la prueba estática se basa en evaluación manual de los productos de trabajo (es decir, revisiones) o en la evaluación basada en herramientas del código u otros productos de trabajo (es decir, el análisis estático). Ambos tipos de prueba estática evalúan el código u otro producto de trabajo que se esté probando sin ejecutar, de forma efectiva, el código o el producto de trabajo que se esté probando.

El es importante para los sistemas informáticos de seguridad crítica (por ejemplo, aeronáuticos, médicos o nucleares), pero el análisis estático también se ha vuelto importante y común en otros contextos. Por ejemplo, el análisis estático es una parte importante en la prueba de seguridad. El análisis estático también se incorpora, con frecuencia, a los sistemas de construcción y entrega automatizados, por ejemplo, en el desarrollo Ágil, la entrega continua y el despliegue continuo.

3.1.1 Productos de Trabajo que Pueden Ser Evaluados por una Prueba Estática

Casi cualquier producto de trabajo puede ser examinado mediante una prueba estática (revisiones y/o análisis estático), por ejemplo:

- Especificaciones, incluidos requisitos de negocio, requisitos funcionales y requisitos de seguridad.
- Épicas, historias de usuarios y criterios de aceptación.
- Especificaciones de arquitectura y diseño.
- Código.
- Productos de prueba, incluyendo planes de prueba, casos de prueba, procedimientos de prueba y guiones de prueba automatizados.
- Guías de usuario.
- Páginas web.
- Contratos, planes de proyecto, calendarios y presupuestos.
- Modelos, tales como diagramas de actividad, que pueden ser usados para la prueba basada en modelos (ver el Programa de Estudio de la Extensión del Nivel Básico de Probador de Prueba Basado en Modelos (ISTQB-MBT) y Kramer 2016).

Las revisiones se pueden aplicar a cualquier producto de trabajo que los participantes sepan cómo leer y comprender. El análisis estático puede ser aplicado eficientemente a cualquier producto de trabajo con una estructura formal (típicamente código o modelos) para el cual exista una herramienta de análisis estático apropiada. El análisis estático, incluso, se puede aplicar con herramientas que evalúan los productos de trabajo escritos en lenguaje natural, como los requisitos (por ejemplo, la corrección ortográfica, la gramática y la legibilidad).

3.1.2 Ventajas de la Prueba Estática

Las técnicas de prueba estática aportan una serie de ventajas. Cuando se aplica al principio del ciclo de vida del desarrollo de software, la prueba estática permite la detección temprana de defectos antes de que se realicen pruebas dinámicas (por ejemplo, en revisiones de requisitos o especificaciones de diseño, refinamiento de la cartera del producto, etc.). Los defectos que se detectan de forma temprana suelen ser mucho más baratos de eliminar que los que se detectan más tarde en el ciclo de vida, especialmente si se comparan con los defectos que se detectan después del despliegue del software y durante el uso activo del mismo. El uso de técnicas de prueba estática para detectar defectos y luego corregirlos de forma

temprana es, casi siempre, mucho más barato para la organización que el uso de pruebas dinámicas para encontrar defectos en el objeto de prueba y luego corregirlos, especialmente cuando se consideran los costes adicionales asociados con la actualización de otros productos de trabajo y la realización de pruebas de confirmación y regresión.

Ventajas adicionales de la prueba estática pueden incluir:

- Detección y corrección de defectos de forma más eficiente y antes de la ejecución de la prueba dinámica.
- Identificar defectos que no se encuentran fácilmente mediante prueba dinámica.
- Prevenir defectos en el diseño o la codificación descubriendo inconsistencias, ambigüedades, contradicciones, omisiones, inexactitudes y redundancias en los requisitos.
- Incrementar la productividad de desarrollo (por ejemplo, debido a un diseño mejorado, código más mantenible).
- Reducir el coste y el tiempo de desarrollo.
- Reducir el coste y el tiempo de la prueba.
- Reducir el coste total de la calidad durante la vida útil del software, debido a la reducción de los fallos en etapas posteriores del ciclo de vida o después de la entrega en operación.
- Mejorar la comunicación entre los miembros del equipo en el curso de la participación en revisiones.

3.1.3 Diferencias entre la Prueba Estática y la Prueba Dinámica

La prueba estática y la prueba dinámica pueden tener los mismos objetivos (véase la sección 1.1.1), tales como proporcionar una evaluación de la calidad de los productos de trabajo e identificar los defectos tan temprano como sea posible. La prueba estática y dinámica se complementan entre sí al encontrar diferentes tipos de defectos.

Una de las principales diferencias es que la prueba estática detecta defectos en los productos de trabajo directamente, en lugar de identificar los fallos causados por defectos cuando se ejecuta el software. Un defecto puede residir en un producto de trabajo durante mucho tiempo sin provocar un fallo. El camino donde se encuentra el defecto puede ser practicado con poca frecuencia o difícil de alcanzar, por lo que no será fácil construir y ejecutar una prueba dinámica que lo detecte. La prueba estática puede ser capaz de encontrar el defecto con un esfuerzo mucho menor.

Otra diferencia es que la prueba estática se puede utilizar para mejorar la consistencia y la calidad interna de los productos de trabajo, mientras que la prueba dinámica se concentra, normalmente, en los comportamientos visibles desde el exterior.

En comparación con las pruebas dinámicas, los defectos típicos que son más fáciles y económicos de detectar y corregir a través de la prueba estática incluyen:

- Defectos en los requisitos (por ejemplo, inconsistencias, ambigüedades, contradicciones, omisiones, inexactitudes y redundancias).
- Defectos de diseño (por ejemplo, algoritmos o estructuras de base de datos ineficientes, alto acoplamiento, baja cohesión).
- Defectos de codificación (por ejemplo, variables con valores no definidos, variables que nunca se utilizan, código inalcanzable, código duplicado).
- Desviaciones con respecto a estándares (por ejemplo, falta de adhesión a los estándares de codificación).
- Especificaciones de interfaz incorrectas (por ejemplo, unidades de medida diferentes utilizadas por el sistema que realiza la llamada respecto del utilizado por el sistema llamado).
- Vulnerabilidades de seguridad (por ejemplo, susceptibilidad a desbordamientos de la memoria intermedia[27]).
- Deficiencias[28] o inexactitudes en la trazabilidad o cobertura de la base de prueba (por ejemplo, la falta de pruebas para un criterio de aceptación).

Además, la mayoría de los tipos de defectos de mantenibilidad sólo se pueden detectar mediante la prueba estática (por ejemplo, modularización inadecuada, mala reutilización de los componentes, código difícil de analizar y modificar sin introducir nuevos defectos).

3.2 Proceso de Revisión

Las revisiones varían desde informales hasta formales. Las revisiones informales se caracterizan por no seguir un proceso definido y no tener una salida documentada formal. Las revisiones formales se caracterizan por contar con la participación de un equipo, resultados de la revisión documentados y procedimientos documentados para llevar a cabo la revisión. El grado de formalidad de un proceso de revisión está relacionado con factores como el modelo de ciclo de vida de desarrollo de software, la madurez del proceso de desarrollo, la complejidad del producto de trabajo que se debe revisar, cualquier requisito legal o reglamentario, y/o la necesidad de un rastro de auditoría.

El foco de atención de una revisión depende de los objetivos acordados de la misma (por ejemplo, encontrar defectos, lograr comprensión, instruir a los participantes, como los evaluadores y los nuevos miembros del equipo, o discutir y decidir por consenso).

El estándar ISO (ISO/IEC 20246) contiene descripciones más detalladas del proceso de revisión de productos de trabajo, incluyendo roles y técnicas de revisión.

[27] **Consultar ****
[28] **Consultar ****

3.2.1 Proceso de Revisión de Productos de Trabajo

El proceso de revisión comprende las siguientes actividades principales:

Planificar

- Definir el alcance, que incluye el objetivo de la revisión, qué documentos o partes de documentos se deben revisar y las características de calidad que se deben evaluar.
- Estimar el esfuerzo y plazos[29].
- Identificar las características de la revisión, tales como el tipo de revisión con los roles, las actividades y las listas de comprobación.
- Seleccionar las personas que participarán en la revisión y asignación de roles.
- Definir los criterios de entrada y salida para tipos de revisión más formales (por ejemplo, inspecciones).
- Comprobar el cumplimiento de los criterios de entrada (para tipos de revisión más formales).

Iniciar Revisión

- Distribuir el producto de trabajo (físicamente o por medios electrónicos) y otros materiales, tales como formularios de registro de cuestiones, listas de comprobación y productos de trabajo relacionados.
- Explicar a los participantes el alcance, los objetivos, el proceso, las funciones y los productos del 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.
- Notificar posibles defectos, recomendaciones y preguntas.

Comunicar y Analizar Cuestiones

- Comunicar los defectos potenciales identificados (por ejemplo, en una reunión de revisión).
- Analizar los posibles defectos, asignar la propiedad[30] y el estado a los mismos.
- Evaluar y documentar las características de calidad.
- Evaluar los hallazgos de la revisión con respecto a los criterios de salida para tomar una decisión de revisión (rechazar; se necesitan cambios importantes; aceptar, posiblemente con cambios menores).

[29] **Consultar ****
[30] **Consultar ****

Corregir e Informar

- Elaborar informes de defecto para aquellos hallazgos que requieren modificaciones.
- Corregir los defectos detectados (en general, realizados por el autor) en el producto de trabajo revisado.
- Comunicar defectos a la persona o equipo adecuado (cuando se encuentran en un producto de trabajo relacionado con el producto de trabajo revisado).
- Registrar el estado actualizado de los defectos (en revisiones formales), incluyendo, eventualmente, el acuerdo del autor del comentario.
- Recopilación de métricas (para tipos de revisión más formales).
- Comprobar el cumplimiento de 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, dependiendo del tipo de revisión y de la formalidad, como se describe en la sección 3.2.3.

3.2.2 Roles y Responsabilidades en una Revisión Formal

Una revisión formal típica incluirá los siguientes roles:

Autor

- Crea el producto de trabajo bajo revisión.
- Corrige los defectos en el producto de trabajo bajo revisión (si fuera necesario).

Dirección

- Es responsable de la planificación de la revisión.
- Decide acerca de la ejecución de las revisiones.
- Asigna personal, presupuesto y tiempo.
- Supervisa la rentabilidad[31] en curso.
- Ejecuta las decisiones de control en caso de resultados inadecuados.

Facilitador (a menudo llamado moderador)

- Asegura el funcionamiento efectivo de las reuniones de revisión (cuando se celebran).
- Si fuera necesario, realiza una mediación entre los distintos puntos de vista.
- A menudo, es la persona de la que depende el éxito de la revisión.

Líder de Revisión

- Asume la responsabilidad general de la revisión.
- Decide quiénes estarán involucrados y organiza cuándo y dónde se llevará a cabo.

[31] **Consultar ****

Revisores

- Pueden ser expertos en la materia, personas que trabajan en el proyecto, implicados con un interés en el producto de trabajo, y/o personas con antecedentes técnicos o de negocio específicos.
- Identifican posibles defectos en el producto de trabajo bajo revisión.
- Pueden representar diferentes perspectivas (por ejemplo, probador, programador, usuario, operador, analista de negocio, experto en usabilidad, etc.).

Escriba[32] (o grabador)

- Recopila[33] los posibles defectos encontrados durante la actividad de revisión individual.
- Registra nuevos defectos potenciales, puntos abiertos y decisiones de la reunión de revisión (cuando se celebra).

En algunos tipos de revisión, una persona puede asumir más de un rol, y las acciones asociadas con cada rol también pueden variar según el tipo de revisión. Además, con la introducción de herramientas para apoyar el proceso de revisión, especialmente el registro de defectos, puntos pendientes y decisiones, a menudo no hay necesidad de un escriba.

Además, son posibles roles más detallados, como se describe en el estándar ISO (ISO/IEC 20246).

3.2.3 Tipos de Revisión

Aunque las revisiones pueden utilizarse para diversos fines, uno de los principales objetivos 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 del negocio y la cultura de la empresa, entre otros criterios de selección.

Las revisiones se pueden clasificar de acuerdo a diversas características. A continuación se enumeran los cuatro tipos más comunes de revisiones y sus características asociadas.

Revisión Informal (por ejemplo, comprobación entre amigos[34], emparejamiento[35], revisión de pares)

- Objetivo principal: detectar defectos potenciales.
- Posibles objetivos adicionales: generar nuevas ideas o soluciones, resolver de forma rápida problemas menores.
- No se basa en un proceso formal (documentado).
- Puede no implicar una reunión de revisión.
- Puede ser realizado por un compañero de trabajo del autor (comprobación entre amigos) o por más personas.
- Se pueden documentar los resultados.
- Varía en utilidad dependiendo de los revisores.
- El uso de listas de comprobación es opcional.

[32] **Consultar ****
[33] **Consultar ****
[34] **Consultar ****
[35] **Consultar ****

- Utilizado con mucha frecuencia en el desarrollo Ágil.

Revisión Guiada

- Objetivos principales: detectar defectos, mejorar el producto software, considerar implementaciones alternativas, evaluar la conformidad con estándares y especificaciones.
- Posibles objetivos adicionales: intercambio de ideas sobre técnicas o variaciones de estilo, formación de los participantes, alcanzar un consenso.
- La preparación individual antes de la reunión de revisión es opcional.
- Normalmente, la reunión de revisión está dirigida por el autor del producto de trabajo.
- El escriba es obligatorio.
- El uso de listas de comprobación es opcional.
- Puede tomar la forma de escenarios, ensayos[36], o simulaciones.
- Se pueden elaborar registros de posibles defectos e informes de revisión.
- En la práctica puede variar de bastante informal a muy formal.

Revisión Técnica

- Objetivos principales: lograr un consenso, detectar defectos potenciales.
- Posibles objetivos adicionales: evaluar la calidad y generar confianza en el producto de trabajo, generar nuevas ideas, motivar y capacitar a los autores para mejorar los futuros productos de trabajo, considerar implementaciones alternativas.
- Los revisores deben ser pares técnicos del autor y expertos técnicos en la misma u otras disciplinas.
- Es necesario que haya una preparación individual antes de la reunión de revisión.
- La reunión de revisión es opcional, lo ideal es que la dirija un facilitador capacitado (normalmente no el autor).
- El escriba es obligatorio, lo ideal es que no sea el autor.
- El uso de listas de comprobación es opcional.
- Normalmente se elaboran registros de defectos potenciales e informes de revisión.

[36] **Consultar ****

Inspección

- Objetivos principales: detectar defectos potenciales, evaluar la calidad y generar confianza en el producto de trabajo, prevenir futuros defectos similares mediante el aprendizaje del autor y el análisis de la causa raíz.
- Posibles objetivos adicionales: motivar y capacitar a los autores para que mejoren los futuros productos de trabajo y el proceso de desarrollo de software, alcanzar un 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 necesario que haya una preparación individual antes de la reunión de revisión.
- Los revisores son pares del autor o expertos en otras disciplinas que son relevantes para el producto de trabajo.
- Se utilizan criterios especificados de entrada y salida.
- El escriba es obligatorio.
- La reunión de revisión es dirigida por un facilitador capacitado (no el autor).
- El autor no puede actuar como líder de revisión, lector o escriba.
- Se elaboran registros de defectos potenciales e informes de revisión.
- Se recopilan y utilizan métricas para mejorar el proceso de desarrollo de software completo, incluido el proceso de inspección.

Un mismo 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 asegurar que el producto de trabajo esté listo para una revisión técnica.

Los tipos de revisión descritos anteriormente se pueden realizar como revisiones entre pares, es decir, por profesionales con un nivel aproximado similar en la organización.

Los tipos de defectos detectados en una revisión varían, dependiendo especialmente del producto de trabajo sujeto a revisión. Véase la sección 3.1.3 para ejemplos de defectos que se pueden detectar en las revisiones de diferentes productos de trabajo, y véase Gilb 1993 para información sobre inspecciones formales.

3.2.4 Aplicación de Técnicas de Revisión

Hay diversas técnicas de revisión que pueden aplicarse durante la actividad de revisión individual (es decir, la preparación individual) para detectar defectos. Estas técnicas se pueden utilizar en todos 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 diversos tipos de revisión.

Ad hoc[37]

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 las cuestiones a medida que los encuentran. La revisión ad hoc es una técnica utilizada con frecuencia que requiere poca preparación. Esta técnica depende en gran medida de las competencias de los revisores y puede llevar a que diferentes revisores informen de muchas cuestiones duplicadas.

Basada en Lista de Comprobación

Una revisión basada en listas de comprobación es una técnica sistemática, en la cual los revisores detectan cuestiones basadas en listas de comprobación que se distribuyen al inicio de la revisión (por ejemplo, por el facilitador). Una lista de comprobación de revisión 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 está revisando y deben mantenerse regularmente para cubrir los tipos de cuestiones que no se han tenido en cuenta en revisiones anteriores. La principal ventaja de la técnica basada en listas de comprobación es la cobertura sistemática de los tipos de defectos típicos. Se debe tener cuidado no sólo de, simplemente, seguir la lista de comprobación en la revisión individual, sino también de buscar defectos fuera de la lista de comprobación.

Escenarios y Ensayos

En una revisión basada en escenarios, los revisores reciben pautas estructuradas sobre cómo leer el producto de trabajo. Un enfoque basado en escenarios ayuda a los revisores a realizar "ensayos" sobre el producto de trabajo basándose en el 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 una lista de comprobación. Al igual que con las revisiones basadas en listas de comprobación, para no pasar por alto otros tipos de defectos (por ejemplo, las prestaciones ausentes), los revisores no deben limitarse a los escenarios documentados.

Basada en Roles

Una revisión basada en roles es una técnica en la cual los revisores evalúan el producto de trabajo desde la perspectiva de roles individuales de los implicados. Los roles típicos incluyen tipos específicos de usuarios finales (experimentados, inexpertos, mayores, niños, etc.), y roles específicos en la organización (administrador de usuarios, administrador de sistemas, probador del rendimiento, etc.).

[37] **Consultar ****

Basada en Perspectiva

En la lectura basada en la perspectiva, similar a una revisión basada en roles, los revisores adoptan los diferentes puntos de vista de los implicados en la revisión individual. Los puntos de vista habituales de los implicados incluyen el usuario final, marketing, diseñador, probador u operaciones. El uso de diferentes puntos de vista de los implicados conduce a una mayor profundidad en la revisión individual con menos duplicación de cuestiones entre los revisores.

Además, la lectura basada en la perspectiva también requiere que los revisores intenten utilizar el producto de trabajo bajo revisión para generar el producto que se derivaría a partir de él. Por ejemplo, un probador intentaría generar un borrador de prueba de aceptación si realizara una lectura basada en la perspectiva de una especificación de requisitos para ver si se incluye toda la información necesaria. Además, en la lectura basada en la perspectiva, se espera que se utilicen listas de comprobación.

Estudios empíricos han demostrado que la lectura basada en la perspectiva es la técnica general más efectiva para revisar los requisitos y los productos de trabajo técnicos. Un factor clave del éxito es incluir y sopesar[38] adecuadamente los diferentes puntos de vista de los implicados, en función del riesgo. Ver Shul 2000 para detalles sobre la lectura basada en la perspectiva, y Sauer 2000 para la efectividad de los diferentes tipos de revisión.

3.2.5 Factores de Éxito para las Revisiones

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

Los factores de éxito relativos a la organización 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 adecuados para lograr los objetivos y que son apropiados para el tipo y nivel de los productos de trabajo de software y de los 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 va a revisar.
- Cualquier lista de comprobación utilizada aborda los riesgos principales y se encuentra actualizada.
- Los documentos de gran tamaño se redactan y revisan en pequeños fragmentos, de modo que el control de calidad se ejerce proporcionando a los autores una retroalimentación[39] temprana y frecuente sobre los defectos.
- Los participantes disponen de tiempo suficiente para prepararse.
- Se establece un calendario[40] de revisiones con la debida antelación.
- La dirección[41] apoya el proceso de revisión (por ejemplo, incorporando tiempo suficiente para las actividades de revisión en el calendario de los proyectos).

[38] **Consultar ****
[39] **Consultar ****
[40] **Consultar ****
[41] **Consultar ****

Los factores de éxito relativos a las personas para las revisiones incluyen:

- Se involucra a las personas adecuadas para alcanzar los objetivos de la revisión, por ejemplo, personas con diferentes competencias o perspectivas, que pueden utilizar el documento como una entrada del trabajo.
- Los probadores son vistos como revisores valiosos que contribuyen a la revisión y aprenden sobre el producto de trabajo, lo que les permite preparar pruebas más efectivas y preparar esas pruebas antes.
- Los participantes dedican tiempo y atención adecuados a los detalles.
- Las revisiones se llevan a cabo en pequeños fragmentos, para que los revisores no pierdan la concentración durante la revisión individual y/o la reunión de revisión (cuando se lleva a cabo).
- Los defectos detectados son reconocidos, valorados y tratados objetivamente.
- La reunión está bien gestionada, por lo que los participantes la consideran un uso valioso de su tiempo.
- La revisión se lleva a cabo en un clima de confianza; el resultado no se utilizará para la evaluación de los participantes.
- Los participantes evitan el lenguaje corporal y las conductas que podrían indicar aburrimiento, exasperación u hostilidad hacia otros participantes.
- Se proporciona la formación adecuada, especialmente para los tipos de revisión más formales, como las inspecciones.
- Se promueve una cultura de aprendizaje y mejora de los procesos.

Ver Gilb 1993, Wiegers 2002, y van Veenendaal 2004 para más información sobre el éxito en las revisiones.

4 Técnicas de Prueba

Duración: 330 minutos

Palabras Clave

análisis de valores frontera ("boundary value analysis")
cobertura ("coverage")
cobertura de decisión ("decision coverage")
cobertura de sentencia ("statement coverage")
partición de equivalencia ("equivalence partitioning")
predicción de errores ("error guessing")
prueba basada en listas de comprobación ("checklist-based testing")
prueba de caso de uso ("use case testing")
prueba de tabla de decisión ("decision table testing")
prueba de transición de estado ("state transition testing")
prueba exploratoria ("exploratory testing")
técnica de prueba ("test technique")
técnica de prueba basada en la experiencia ("experience-based test technique")
técnica de prueba de caja blanca ("white-box test technique")
técnica de prueba de caja negra ("black-box test technique")

Objetivos de Aprendizaje para Técnicas de Prueba

4.1 Categorías de Técnicas de Prueba

NB-4.1.1 (K2) Explicar las características, los aspectos comunes y las diferencias entre las técnicas de prueba de caja negra, las técnicas de prueba de caja blanca y las técnicas de prueba basadas en la experiencia.

4.2 Técnicas de Prueba de Caja Negra

NB-4.2.1 (K3) Aplicar la técnica de partición de equivalencia para obtener casos de prueba a partir de requisitos especificados.

NB-4.2.2 (K3) Aplicar la técnica de análisis de valores frontera para obtener casos de prueba a partir de los requisitos especificados.

NB-4.2.3 (K3) Aplicar la técnica de tablas de decisión para obtener casos de prueba a partir de requisitos especificados.

NB-4.2.4 (K3) Aplicar la técnica de prueba de transición de estado para obtener casos de prueba a partir de requisitos especificados.

NB-4.2.5 (K2) Explicar cómo obtener casos de prueba a partir de un caso de uso.

4.3 Técnicas de Prueba de Caja Blanca

NB-4.3.1 (K2) Explicar la cobertura de sentencia.

NB-4.3.2 (K2) Explicar la cobertura de decisión.

NB-4.3.3 (K2) Explique el valor de la cobertura de sentencia y la cobertura de decisión.

4.4 Técnicas de Prueba Basadas en la Experiencia

NB-4.4.1 (K2) Explicar la predicción de errores.

NB-4.4.2 (K2) Explicar la prueba exploratoria.

NB-4.4.3 (K2) Explicar la prueba basada en una lista de comprobación.

4.1 Categorías de Técnicas de Prueba

El objetivo de una técnica de prueba, incluidas las que se analizan en esta sección, es ayudar a identificar las condiciones de prueba, los casos de prueba y los datos de prueba.

4.1.1 Elección de Técnicas de Prueba

La elección de qué técnicas de prueba se van a utilizar depende de una serie de factores, entre los que se incluyen los siguientes:

- Tipo de componente o sistema.
- Complejidad del componente o del sistema.
- Estándares de regulación.
- Requisitos del cliente o contractuales.
- Niveles de riesgo.
- Clases de riesgo.
- Objetivos de prueba.
- Documentación disponible.
- Conocimientos y competencias del probador.
- Herramientas disponibles.
- Tiempo y presupuesto.
- Modelo de ciclo de vida de desarrollo de software.
- Uso previsto del software.
- Experiencia previa en el uso de las técnicas de prueba en el componente o sistema que se va a probar.
- Los tipos de defectos esperados en el componente o sistema.

Algunas técnicas son más adecuadas para ciertas situaciones y niveles de prueba; otras se pueden aplicar en todos los niveles de prueba. En general, cuando se crean casos de prueba, los probadores utilizan una combinación de técnicas de prueba para lograr los mejores resultados a partir del esfuerzo de prueba.

El uso de técnicas de prueba en el análisis de la prueba, el diseño de prueba y las actividades de implementación de la prueba puede variar desde muy informal (poca o ninguna documentación) hasta muy formal. El nivel de formalidad adecuado depende del contexto de la prueba, incluyendo la madurez de los procesos de prueba y desarrollo, las limitaciones de tiempo, los requisitos de seguridad o normativos, los conocimientos y competencias de las personas involucradas y el modelo de ciclo de vida de desarrollo de software que se sigue.

4.1.2 Categorías de Técnicas de Prueba y sus Características

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

Las técnicas de prueba de caja negra (también llamadas técnicas conductuales o basadas en el comportamiento) se basan en un análisis de la base de prueba adecuada (por ejemplo, documentos de requisitos formales, especificaciones, casos de uso, historias de usuarios o procesos de negocio). Estas técnicas son aplicables tanto a la prueba funcional como a la no funcional. Las técnicas de prueba de caja negra se concentran en las entradas y salidas del objeto de prueba sin 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, el diseño detallado, la 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 desarrolladores, probadores y usuarios para diseñar, implementar y ejecutar pruebas. A menudo, estas técnicas se combinan con técnicas de prueba de caja negra y caja blanca.

Entre las características comunes de las técnicas de prueba de la caja negra se incluyen las siguientes:

- Las condiciones de prueba, casos de prueba y datos de prueba se deducen 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 utilizar para detectar diferencias[42] entre los requisitos y la implementación de los requisitos, así como desviaciones con respecto a los requisitos.
- La cobertura se mide en función de los elementos probados en la base de prueba y de la técnica aplicada a la base de prueba.

Entre las características comunes de las técnicas de prueba de caja blanca se incluyen las siguientes:

- Las condiciones de la prueba, los casos de prueba y los datos de la prueba se deducen de una base de prueba que puede incluir el código, la arquitectura del software, el diseño detallado o cualquier otra fuente de información relacionada con la estructura del software.
- La cobertura se mide en base a los elementos probados dentro de una estructura seleccionada (por ejemplo, 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.

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

- Las condiciones de prueba, los casos de prueba y los datos de prueba se deducen de una base de prueba que puede incluir el conocimiento y la experiencia de probadores, desarrolladores, usuarios y otros implicados.

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

[42] **Consultar ****

El estándar internacional (ISO/IEC/IEEE 29119-4) contiene descripciones de técnicas de prueba y sus correspondientes medidas de cobertura (ver Craig 2002 y Copeland 2004 para más información sobre técnicas).

4.2 Técnicas de Prueba de Caja Negra

4.2.1 Partición de Equivalencia

La partición de equivalencia divide los datos en particiones (también conocidas como clases de equivalencia) de tal manera que se espera que todos los miembros de una partición dada sean procesados de la misma manera (ver Kaner 2013 y Jorgensen 2014). Existen particiones de equivalencia tanto para valores válidos como no válidos.

- Los valores válidos son los valores que debe aceptar el componente o el sistema. Una partición de equivalencia que contiene valores válidos se llama "partición de equivalencia válida".
- Los valores no válidos son valores que deben ser rechazados por el componente o sistema. Una partición de equivalencia que contiene valores no válidos se llama "partición de equivalencia no válida".
- Las particiones pueden identificarse para cualquier elemento de datos relacionado con el objeto de prueba, incluyendo entradas, salidas, valores internos, valores relacionados con el tiempo (por ejemplo, antes o después de un evento) y para parámetros de interfaz (por ejemplo, componentes integrados que se están probando durante la prueba de integración).
- Cualquier partición se puede dividir en subparticiones si fuera necesario.
- Cada valor debe pertenecer a una y sólo a una partición de equivalencia.
- Cuando se utilizan particiones de equivalencia no válidas en casos de prueba, deben probarse individualmente, es decir, no combinarse con otras particiones de equivalencia no válidas, para garantizar que no se produzca un enmascaramiento de los fallos. Los fallos se pueden enmascarar cuando se producen varios fallos al mismo tiempo, pero sólo uno de ellos es visible, lo que hace que los otros fallos queden sin detectar.

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

4.2.2 Análisis de Valores Frontera

El análisis de valores frontera (AVF) es una extensión de la partición de equivalencia, pero sólo se puede utilizar cuando la partición está ordenada, y consiste en datos numéricos o secuenciales. Los valores mínimo y máximo (o valores inicial y final) de una partición son sus valores frontera (Beizer 1990).

Por ejemplo, suponer que un campo de entrada acepta un único valor entero como entrada, utilizando un teclado numérico para limitar las entradas de modo que las entradas no enteras sean imposibles. El rango válido es de 1 a 5, ambos valores incluidos. Por lo tanto, hay tres particiones de equivalencia: inválida (demasiado baja); válida; inválida (demasiado alta). Para la partición de equivalencia válida, los valores frontera son 1 y 5. Para la partición no válida (demasiado alta), los valores frontera son 6 y 9. Para la partición inválida (demasiado baja), sólo hay un valor frontera, 0, porque se trata de una partición con un solo miembro.

En el ejemplo anterior, se identifican dos valores frontera por frontera. La frontera entre inválido (demasiado bajo) y válido proporciona los valores de prueba 0 y 1. La frontera entre válido e inválido (demasiado alto) proporciona los valores de prueba 5 y 6. Algunas variantes de esta técnica identifican tres valores frontera por frontera: los valores antes, en y justo después de la frontera. En el ejemplo anterior, utilizando tres puntos para los valores frontera, los valores de la prueba de la frontera inferior son 0, 1 y 2, y los valores de la prueba de la frontera superior son 4, 5 y 6 (Jorgensen 2014).

El comportamiento en las fronteras de las particiones de equivalencia es más probable que sea incorrecto que el comportamiento dentro de las particiones. Es importante recordar que tanto las fronteras especificadas como las implementadas pueden desplazarse a posiciones por encima o por debajo de sus posiciones previstas, pueden omitirse por completo o pueden complementarse con fronteras adicionales no deseadas. El análisis y la prueba del valor frontera revelarán casi todos estos defectos forzando al software a mostrar comportamientos de una partición distinta a la que debería pertenecer el valor frontera.

El análisis de valores frontera se puede aplicar en todos los niveles de prueba. Esta técnica se utiliza generalmente para probar los requisitos que requieren un rango de números (incluyendo fechas y horas). La cobertura de frontera para una partición se mide como el número de valores frontera probados, dividido por el número total de valores frontera de prueba identificados, normalmente expresado como un porcentaje.

4.2.3 Prueba de Tabla de Decisión

Las técnicas de prueba combinatorias son útiles para probar la implementación de requisitos de sistema que especifican cómo diferentes combinaciones de condiciones generan diferentes resultados. Un enfoque para este tipo de prueba es la prueba de tabla de decisión.

Las tablas de decisión son una buena manera de documentar reglas de negocio complejas que un sistema debe implementar. Al crear tablas de decisión, el probador identifica las condiciones (a menudo entradas) y las acciones resultantes (a menudo salidas) del sistema. Éstas conforman 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 a esa regla. Los valores de las condiciones y acciones, normalmente se muestran como valores booleanos (verdadero o falso) o valores discretos (por ejemplo, rojo, verde, azul), pero también pueden ser números o intervalos de números. Estos diferentes tipos de condiciones y acciones pueden estar juntos en la misma tabla.

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

Para las condiciones:

- S[43] significa que la condición es verdadera (también se puede mostrar como V[44] o 1).
- N[45] significa que la condición es falsa (también se puede mostrar como F[46] o 0).
- Un guion "-" significa que el valor de la condición no importa (también puede mostrarse como N/A[47]).

[43] **Consultar ****
[44] **Consultar ****
[45] **Consultar ****
[46] **Consultar ****
[47] **Consultar ****

Para las acciones:

- X significa que la acción debe ocurrir (también puede mostrarse como S o V o 1).
- En blanco significa que la acción no debe ocurrir (también puede mostrarse como - o N o F o 0).

Una tabla de decisión completa[48] tiene suficientes columnas para cubrir cada combinación de condiciones. La tabla se puede colapsar borrando las columnas que contienen combinaciones de condiciones imposibles, las columnas que contienen combinaciones de condiciones posibles pero no factibles y las columnas que prueban combinaciones de condiciones que no afectan al resultado. Para más información sobre cómo colapsar las tablas de decisión, consultar el Programa de Estudio de Nivel Avanzado de Analista de Pruebas del ISTQB (ISTQB-ATA).

La cobertura estándar mínima habitual para la prueba de tabla de decisión es tener al menos un caso de prueba por regla de decisión en la tabla. Esto implica, normalmente, 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 la prueba de tabla de decisión es que ayuda a identificar todas las combinaciones importantes de condiciones, algunas de las cuales, de otra manera, podrían ser ignoradas. También ayuda a encontrar cualquier desfase[49] 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 Prueba de Transición de Estado

Los componentes o sistemas pueden responder de forma diferente a un evento dependiendo de las condiciones del momento o de su historia previa (por ejemplo, los eventos que han ocurrido desde que se inicializó el sistema). La historia previa puede resumirse utilizando el concepto de estado. Un diagrama de transición de estado muestra los posibles estados del software, así como la forma en que el software entra, sale y realiza las transiciones entre estados. Una transición se inicia con un evento (por ejemplo, la entrada de un valor por parte del usuario en un campo). El evento resulta en una transición. Si el mismo evento puede resultar en dos o más transiciones diferentes desde el mismo estado, ese evento puede estar condicionado por una condición de guarda[50]. El cambio de estado puede provocar que el software tome una acción (por ejemplo, emitir el resultado de un cálculo o un mensaje de error).

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

Las pruebas pueden ser diseñadas para cubrir una secuencia de estados típica, para practicar todos los estados, para practicar cada transición, para practicar secuencias específicas de transiciones, o para probar transiciones inválidas.

La prueba de transición de estado se utiliza para aplicaciones basadas en menús y es extensamente utilizada en la industria del software embebido. La técnica también es adecuada para modelar un escenario de negocio con estados específicos o para probar la navegación en pantalla. El concepto de estado es abstracto: puede representar unas pocas líneas de código o todo un proceso de negocio.

[48] **Consultar ****
[49] **Consultar ****
[50] **Consultar ****

La cobertura se mide, habitualmente, como el número de estados o transiciones identificados probados, dividido por el número total de estados o transiciones identificados en el objeto de prueba, normalmente expresado como un porcentaje. Para más información sobre los criterios de cobertura para la prueba de transición de estado, consultar el Programa de Estudio de Nivel Avanzado de Analista de Pruebas del ISTQB (ISTQB-ATA).

4.2.5 Prueba de Caso de Uso

Las pruebas se pueden obtener[51] a partir de casos de uso, que son una forma específica de diseñar interacciones con elementos software, incorporando requisitos para las funciones del 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 sujetos (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 mediante precondiciones, poscondiciones y lenguaje natural cuando resulte adecuado. Las interacciones entre los actores y el sujeto pueden resultar en 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, incluyendo el tratamiento de un comportamiento excepcional y de errores (respuesta del sistema y recuperación de errores de programación, de aplicación y de comunicación, por ejemplo, resultando en un mensaje de error). Las pruebas están diseñadas para practicar las conductas definidas (básicas, excepcionales o alternativas, y tratamiento 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 expresado como un porcentaje.

Para obtener más información sobre los criterios de cobertura para la prueba de caso de uso, consultar el Programa de Estudio de Nivel Avanzado de Analista de Pruebas del ISTQB (ISTQB-ATA).

4.3 Técnicas de Prueba de Caja Blanca

Las técnicas de prueba de caja blanca se basan en la estructura interna del objeto de prueba. Las técnicas de prueba de caja blanca se pueden utilizar en todos los niveles de prueba, pero las dos técnicas relacionadas con el código que se discuten en esta sección se utilizan con mayor frecuencia en el nivel de prueba de componente. Hay 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 en este documento. Para más información sobre estas técnicas, consultar el Programa de Estudio de Analista de Pruebas Técnicas de Nivel Avanzado del ISTQB.

4.3.1 Prueba y Cobertura de Sentencia

La prueba de sentencia practica las sentencias 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.

[51] **Consultar ****

4.3.2 Prueba y Cobertura de Decisión

La prueba de decisión practica las decisiones en el código y prueba el código que se ejecuta basado en los resultados de la decisión. Para ello, los casos de prueba siguen los flujos de control que se producen desde un punto de decisión (por ejemplo, para una declaración IF, uno para el resultado verdadero y otro para el resultado falso; para una declaración CASE, se necesitarían casos de prueba para todos los resultados posibles, 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 la Prueba de Sentencia y Decisión

Cuando se logra una cobertura del 100% de sentencia, se asegura de que todas las sentencias ejecutables del código se han probado al menos una vez, pero no asegura que se haya probado toda la lógica de decisión. De las dos técnicas de caja blanca discutidas en este programa, la prueba de sentencia puede proporcionar menos cobertura que la prueba de decisión.

Cuando se alcanza el 100% de cobertura de decisión, se ejecutan todos los resultados de decisión, lo que incluye probar el resultado verdadero y también el resultado falso, incluso cuando no hay una sentencia falsa explícita (por ejemplo, en el caso de una sentencia IF sin un ELSE en el código). La cobertura de sentencia ayuda a encontrar defectos en el código que no fueron practicados por otras pruebas. La cobertura de decisión ayuda a encontrar defectos en el código donde otras pruebas no han tenido ambos resultados, verdadero y falso.

Lograr una cobertura del 100% de decisión garantiza una cobertura del 100% de sentencia (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 obtienen a partir de la competencia e intuición del probador y de su experiencia con aplicaciones y tecnologías similares. Estas técnicas pueden ser útiles para identificar pruebas que no fueron fácilmente identificadas por otras técnicas más sistemáticas. Dependiendo del enfoque y la experiencia del probador, estas técnicas pueden lograr grados muy diferentes de cobertura y efectividad. La cobertura puede ser difícil de evaluar y puede no ser medible con estas técnicas.

En las siguientes secciones se abordan las técnicas basadas en la experiencia que se utilizan con frecuencia.

4.4.1 Predicción de Errores

La predicción de errores es una técnica utilizada para anticipar la ocurrencia de equivocaciones, defectos y fallos, basada en el conocimiento del probador, incluido:

- Cómo ha funcionado la aplicación en el pasado.
- Qué tipo de equivocaciones tienden a cometer los desarrolladores.
- Fallos que se han producido en otras aplicaciones.

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

4.4.2 Prueba Exploratoria

En la prueba exploratoria, se diseñan, ejecutan, registran y evalúan de forma dinámica pruebas informales (no predefinidas) durante la ejecución de la prueba. Los resultados de la prueba se utilizan para aprender más sobre el componente o sistema, y para crear pruebas para las áreas que pueden necesitar ser probadas con mayor intensidad.

A veces se realiza la prueba exploratoria utilizando la prueba basada en sesiones para estructurar la actividad. En la prueba basada en sesiones, la prueba exploratoria se lleva a cabo dentro de un período de tiempo definido, y el probador utiliza un contrato de prueba que contiene los objetivos de la prueba para guiar la prueba. El probador puede usar hojas de sesión de prueba[52] para documentar los pasos seguidos y los descubrimientos realizados.

La prueba exploratoria es más útil cuando las especificaciones son escasas o inadecuadas o cuando hay una presión significativa con respecto al tiempo para la prueba. La prueba exploratoria también es útil para complementar otras técnicas de prueba más formales.

La prueba exploratoria está fuertemente asociada con las estrategias de prueba reactivas (ver sección 5.2.2). La prueba exploratoria puede incorporar el uso de otras técnicas de caja negra, caja blanca y basadas en la experiencia.

4.4.3 Prueba Basada en Listas de Comprobación

En la prueba basada en listas de comprobación, los probadores diseñan, implementan y ejecutan pruebas para cubrir las condiciones de prueba que se encuentran en una lista de comprobación. Como parte del análisis, los probadores crean una nueva lista de comprobación o amplían una ya existente, pero los probadores también pueden utilizar una lista de comprobación ya existente sin modificar. Estas listas de comprobación pueden elaborarse basándose en la experiencia, el conocimiento de lo que es importante para el usuario o la comprensión de por qué y cómo falla el software.

Se pueden crear listas de comprobación para dar soporte a varios tipos de prueba, incluyendo pruebas funcionales y no funcionales. A falta de casos de prueba detallados, las pruebas basadas en listas de comprobación pueden proporcionar directrices y un cierto grado de consistencia. Dado que se trata de listas de alto nivel, es probable que se produzca cierta variabilidad en las pruebas reales, lo que puede dar lugar a una mayor cobertura pero a una menor repetibilidad.

[52] **Consultar ****

5 Gestión de la Prueba

Duración: 225 minutos

Palabras Clave

control de la prueba ("test control")
criterios de entrada ("entry criteria")
criterios de salida ("exit criteria")
enfoque de prueba ("test approach")
estimación de la prueba ("test estimation")
estrategia de prueba ("test strategy")
gestión de defectos ("defect management")
gestión de la configuración ("configuration management")
informe del avance de la prueba ("test progress report")
informe resumen de prueba ("test summary report")
jefe de prueba ("test manager")
monitorización de la prueba ("test monitoring")
nivel de riesgo ("risk level")
plan de prueba ("test plan")
planificación de prueba ("test planning")
probador ("tester")
prueba basada en el riesgo ("risk-based testing")
riesgo ("risk")
riesgo de producto ("product risk")
riesgo de proyecto ("project risk")

Objetivos de Aprendizaje para Gestión de la Prueba

5.1 Organización de la Prueba

NB-5.1.1 (K2) Explicar las ventajas y desventajas de la prueba independiente.

NB-5.1.2 (K1) Identificar las tareas del jefe de la prueba y del probador.

5.2 Planificación y Estimación de la Prueba

NB-5.2.1 (K2) Resumir el objetivo y el contenido de un plan de prueba.

NB-5.2.2 (K2) Diferenciar entre varias estrategias de prueba.

NB-5.2.3 (K2) Proporcionar ejemplos de posibles criterios de entrada y salida.

NB-5.2.4 (K3) Aplicar conocimientos sobre priorización, y dependencias técnicas y lógicas, para programar la ejecución de la prueba para un conjunto determinado de casos de prueba.

NB-5.2.5 (K1) Identificar los factores que influyen en el esfuerzo asociado a la prueba.

NB-5.2.6 (K2) Explicar la diferencia entre dos técnicas de estimación: la técnica basada en métricas[53] y la técnica basada en expertos[54].

[53] **Consultar ****
[54] **Consultar ****

5.3 Monitorización y Control de la Prueba

NB-5.3.1 (K1) Recordar métricas utilizadas para probar.

NB-5.3.2 (K2) Resumir los objetivos, el contenido y las audiencias de los informes de prueba.

5.4 Gestión de la Configuración

NB-5.4.1 (K2) Resumir cómo la gestión de configuración proporciona soporte a la prueba.

5.5 Riesgos y Prueba

NB-5.5.1 (K1) Definir el nivel de riesgo mediante el uso de probabilidad e impacto.

NB-5.5.2 (K2) Distinguir entre riesgos de proyecto y de producto.

NB-5.5.3 (K2) Describir, utilizando ejemplos, cómo el análisis de riesgo de producto puede influir en la intensidad y el alcance de la prueba.

5.6 Gestión de Defectos

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

5.1 Organización de la Prueba

5.1.1 Prueba Independiente

Las tareas de prueba pueden ser realizadas por personas que desempeñan un rol de prueba específico, o por personas que desempeñan otro rol (por ejemplo, clientes). Un cierto grado de independencia, a menudo, hace que el probador sea más efectivo para encontrar defectos debido a las diferencias entre los sesgos asociados al conocimiento del autor y del probador (ver sección 1.5). Sin embargo, la independencia no es un sustituto de la familiaridad, y los desarrolladores pueden encontrar de forma eficiente muchos defectos en su propio código.

Los grados de independencia en la prueba incluyen los siguientes (desde un bajo nivel de independencia a un alto nivel):

- No hay probadores independientes; la única forma de prueba disponible es que los desarrolladores prueben su propio código.
- Desarrolladores independientes o probadores dentro de los equipos de desarrollo o del equipo del proyecto; esta situación podría ser que desarrolladores prueben los productos de sus compañeros.
- Equipo o grupo de prueba independiente dentro de la organización, que informa a la dirección del proyecto o a la dirección ejecutiva.
- Probadores independientes de la organización de negocio o de la comunidad de usuarios, o con especializaciones en tipos específicos de prueba tales como usabilidad, seguridad, rendimiento, cumplimiento/normativo o portabilidad.
- Probadores independientes externos a la organización, ya sea que trabajen en sus instalaciones (internalización[55]) o fuera de ellas (externalización[56]).

[55] **Consultar ****
[56] **Consultar ****

Para la mayoría de los tipos de proyectos, generalmente es mejor que tengan diferentes niveles de prueba, con algunos de estos niveles tratados por probadores independientes. Los desarrolladores deben participar en la prueba, especialmente en los niveles inferiores, de manera que puedan ejercer control sobre la calidad de su propio trabajo.

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

Los beneficios potenciales de la independencia de la prueba incluyen:

- Es probable que los probadores independientes reconozcan diferentes tipos de fallos en comparación con los desarrolladores debido a sus diferentes contextos, perspectivas técnicas y sesgos.
- Un probador independiente puede verificar, cuestionar o refutar las suposiciones hechas por los implicados durante la especificación e implementación del sistema.

Las posibles desventajas de la independencia de la prueba incluyen:

- Aislamiento con respecto al equipo de desarrollo, lo que conduce a una falta de colaboración, retrasos en la retroalimentación al equipo de desarrollo o una relación de confrontación con el equipo de desarrollo.
- Los desarrolladores pueden perder el sentido de la responsabilidad con respecto a la calidad.
- Los probadores independientes pueden ser vistos como un cuello de botella o ser culpados por los retrasos en el lanzamiento o liberación.
- Los probadores independientes pueden carecer de información importante (por ejemplo, sobre el objeto de prueba).

Muchas organizaciones son capaces de lograr con éxito los beneficios de la independencia de la prueba, evitando al mismo tiempo los inconvenientes.

5.1.2 Tareas de un Jefe de Prueba y un Probador

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

El jefe de la prueba asume la responsabilidad general del proceso de la prueba y el éxito en el liderazgo de las actividades de la prueba. El rol de gestión de la prueba puede ser desempeñado por un jefe de prueba profesional, o por un jefe de proyecto, un jefe de desarrollo o un responsable de aseguramiento de la calidad. En proyectos u organizaciones más grandes, varios equipos de prueba pueden informar a un jefe de prueba, a un entrenador de prueba o a un coordinador de pruebas, cada equipo está encabezado por un líder de prueba o un probador líder.

Las tareas habituales 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 de los tiempos, esfuerzos y costes de la prueba, la adquisición de recursos, la definición de niveles y ciclos de prueba y la planificación de la gestión de defectos.
- Redactar y actualizar el/los plan(es) de prueba.
- Coordinar el/los plan(es) de prueba con los jefes de proyecto, los propietarios de producto y otros.
- Compartir las perspectivas con respecto a las pruebas con otras actividades del proyecto, como la planificación de la integración.
- Iniciar el análisis, diseño, implementación y ejecución de pruebas, monitorizar el avance y los resultados de la prueba y comprobar el estado de los criterios de salida (o definición de hecho).
- Preparar y entregar informes de avance de la prueba e informes de resumen de prueba basados en la información recopilada.
- Adaptar la planificación en función de los resultados y avance de la prueba (a veces documentados en informes de avance de la prueba y/o en informes de resumen de prueba para otras pruebas ya realizadas en el proyecto) y tomar las medidas necesarias para el control de la prueba.
- Apoyar la implantación del sistema de gestión de defectos y una gestión de la configuración adecuada para los productos de prueba.
- Introducir métricas adecuadas para medir el avance de la prueba y evaluar la calidad de la prueba y del producto.
- Apoyar la selección e implementación de herramientas para dar soporte al proceso de prueba, incluyendo la recomendación del presupuesto para la selección de herramientas (y posiblemente la compra y/o el soporte), la asignación de tiempo y esfuerzo para los proyectos piloto, y la provisión de un soporte continuo en el uso de la herramienta o herramientas.
- Decidir sobre la implementación de los entornos de prueba (uno o más de uno).
- Promover y defender a los probadores, al equipo de prueba y a la profesión de prueba dentro de la organización.
- Desarrollar las competencias y las carreras profesionales de los probadores (por ejemplo, mediante planes de formación, evaluaciones del desempeño, entrenamiento[57], etc.).

La forma en que se lleva a cabo el rol de jefe de la prueba varía en función del ciclo de vida de desarrollo de software. Por ejemplo, en el desarrollo Ágil, algunas de las tareas mencionadas anteriormente son tratadas por el equipo Ágil, especialmente aquellas tareas relacionadas con la prueba diaria realizadas dentro del equipo, a menudo por un probador que trabaja dentro del equipo. Algunas de las tareas que abarcan múltiples equipos o toda la organización, o que tienen que ver con la gestión de personal, pueden ser realizadas por jefes de prueba externos al equipo de desarrollo, a los que a veces se les llama orientadores de prueba. Para obtener más información sobre la gestión del proceso de prueba, consultar Black 2009.

[57] **Consultar ****

Las tareas habituales del probador pueden incluir:

- Revisar y contribuir a los planes de prueba.
- Analizar, revisar y evaluar los requisitos, historias de usuario y criterios de aceptación, especificaciones y modelos en lo que respecta a la capacidad de ser probado (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, preparar y verificar los entornos de prueba, a menudo en coordinación con las áreas de administración de sistemas y gestión de redes.
- Diseñar e implementar casos de prueba y procedimientos de prueba.
- Preparar y obtener datos de prueba.
- Crear el calendario de ejecución de prueba detallado.
- Ejecutar pruebas, evaluar los resultados y documentar las desviaciones de los resultados esperados.
- Utilizar herramientas apropiadas para facilitar el proceso de prueba.
- Automatizar las pruebas según sea necesario (puede contar con el apoyo de un desarrollador o de un experto en automatización de pruebas).
- Evaluar características no funcionales tales como eficiencia de desempeño, fiabilidad, usabilidad, seguridad, compatibilidad y portabilidad.
- Revisar pruebas desarrolladas por otros.

Las personas que trabajan en análisis de prueba, diseño de prueba, tipos específicos de prueba o automatización de la prueba pueden ser especialistas en estos roles. Dependiendo de los riesgos asociados al producto y al 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 componente y en el nivel de prueba de integración de componentes, el papel de un probador lo desempeñan, a menudo, los desarrolladores. En el nivel de prueba de aceptación, el papel de un probador suele ser desempeñado por analistas de negocio, expertos en la materia y usuarios. En el nivel de prueba de sistema y en el nivel de prueba de integración de sistemas, el papel de un probador suele ser desempeñado por un equipo de prueba independiente. En el nivel de prueba de aceptación operativa, el papel de un probador suele ser desempeñado por el personal de operaciones y/o administración de sistemas.

5.2 Planificación y Estimación de la Prueba

5.2.1 Propósito y Contenido de un Plan de Prueba

Un plan de prueba describe las actividades de prueba para proyectos de desarrollo y mantenimiento. La planificación depende de la política y la estrategia de prueba de la organización, los ciclos de vida de desarrollo y los métodos utilizados (véase la sección 2.1), el alcance de la prueba, los objetivos, los riesgos, las restricciones, la criticidad, la capacidad de ser probado, y la disponibilidad de los recursos.

A medida que el proyecto y la planificación de la prueba evolucionan, se dispone de más información y se pueden incluir más detalles en el plan de prueba. La planificación de la prueba es una actividad que se realiza de forma continua a lo largo de todo el ciclo de vida del producto. (Se debe tener en cuenta que el

ciclo de vida del producto puede extenderse más allá del alcance del proyecto para incluir la fase de mantenimiento.). La retroalimentación de las actividades de prueba se debe utilizar para detectar cambios en los riesgos, de tal forma que la planificación pueda ser ajustada. La planificación puede documentarse en un plan maestro de prueba y en planes de prueba separados para los niveles de prueba, como las prueba de sistema y la prueba de aceptación, o para tipos de prueba separados, como la prueba de usabilidad y la prueba de rendimiento. Las actividades de planificación de la prueba pueden incluir las siguientes y algunas de ellas se pueden documentar en un plan de prueba:

- Determinar el alcance, los objetivos y los riesgos de la prueba.
- Definir el enfoque general de la prueba.
- Integrar y coordinar las actividades de prueba en las actividades del ciclo de vida del software.
- Tomar decisiones sobre lo que se va a probar, las personas y otros recursos necesarios para realizar las diversas actividades de prueba, y cómo se llevarán a cabo las actividades de prueba.
- Establecer un calendario para las actividades de análisis, diseño, implementación, ejecución y evaluación de la prueba, ya sea en fechas particulares (por ejemplo, en desarrollo secuencial) o en el contexto de cada iteración (por ejemplo, en desarrollo iterativo).
- Selección de métricas para monitorizar y controlar la prueba.
- Elaborar un presupuesto para las actividades de prueba.
- Determinar el nivel de detalle y la estructura de la documentación de la prueba (por ejemplo, proporcionando plantillas o documentos de ejemplo).

El contenido de los planes de prueba varía, y puede extenderse más allá de los temas previamente identificados. Se pueden encontrar ejemplos de planes de prueba en el estándar ISO (ISO/IEC/IEEE 29119-3).

5.2.2 Estrategia y Enfoque de la Prueba

Una estrategia de prueba proporciona una descripción genérica del proceso de prueba, normalmente a nivel de producto u organización. Entre los tipos comunes de estrategias de prueba se incluyen:

- **Analítica:** Este tipo de estrategia de prueba se basa en el análisis de algún factor (por ejemplo, requisitos o riesgos). Las pruebas basadas en el riesgo son un ejemplo de un enfoque analítico, en el que las pruebas se diseñan y priorizan en función del nivel de riesgo.
- **Basada en Modelos:** En este tipo de estrategia de prueba, las pruebas se diseñan basándose en algún modelo de algún aspecto requerido del producto, como una función, un proceso de negocio, una estructura interna o una característica no funcional (por ejemplo, la fiabilidad). Entre los ejemplos de estos modelos se incluyen los modelos de procesos de negocio, modelos de estado y modelos de crecimiento de la fiabilidad.
- **Metódica:** Este tipo de estrategia de prueba se basa en el uso sistemático de un conjunto predefinido de pruebas o condiciones de prueba, como una taxonomía de los tipos de fallos comunes o probables, una lista de características de calidad importantes o estándares de apariencia corporativa[58] para aplicaciones móviles o páginas web.

[58] **Consultar ****

- **Conforme a Proceso** (o conforme a estándar): Este tipo de estrategia de prueba implica el análisis, diseño e implementación de pruebas basadas en reglas y estándares externos, tales como aquellos especificados por estándares específicos de la industria, por documentación de procesos, por la identificación y uso rigurosos de la base de prueba, o por cualquier proceso o estándar impuesto a o por la organización.
- **Dirigida** (o consultiva): Este tipo de estrategia de prueba se basa principalmente en el asesoramiento, la orientación o las instrucciones de implicados, expertos en el dominio del negocio o expertos en tecnología, que pueden estar fuera del equipo de prueba o fuera de la propia organización.
- **Adversa 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 de productos de prueba existentes (especialmente casos de prueba y datos de prueba), la automatización extensiva de las pruebas de regresión y los juegos de prueba estándar.
- **Reactiva:** En este tipo de estrategia de prueba, la prueba es reactiva al componente o sistema que se está probando, y a los eventos que ocurren durante la ejecución de la prueba, en lugar de ser planificada con antelación (como lo son las estrategias anteriores). Las pruebas se diseñan e implementan, y pueden ser ejecutadas inmediatamente en respuesta al conocimiento obtenido a partir de los resultados de prueba anteriores. La prueba exploratoria es una técnica común que se utiliza en estrategias reactivas.

A menudo, se crea una estrategia de prueba adecuada combinando varios de estos tipos de estrategias de prueba. Por ejemplo, las pruebas basadas en el riesgo (una estrategia analítica) pueden combinarse con pruebas exploratorias (una estrategia reactiva); se complementan entre sí y pueden lograr pruebas más eficaces cuando se utilizan de forma conjunta.

Mientras que la estrategia de prueba proporciona una descripción genérica del proceso de prueba, el enfoque de prueba adapta la estrategia de prueba a un proyecto o lanzamiento particular. El enfoque de prueba es el punto de partida para seleccionar las técnicas de prueba, niveles de prueba, tipos de prueba, para definir los criterios de entrada y salida (o definición de preparado[59] y definición de hecho, respectivamente). La adaptación de la estrategia se basa en 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 de producto. El enfoque seleccionado depende del contexto y puede considerar factores tales como los riesgos, la seguridad física, los recursos y competencias disponibles, la tecnología, la naturaleza del sistema (por ejemplo, construcción a medida frente a Productos Comerciales de Distribución Masiva o COTS por sus siglas en inglés), los objetivos de la prueba y la normativa.

5.2.3 Criterios de Entrada y Criterios de Salida (Definición de Preparado y Definición de Hecho)

Para ejercer un control efectivo sobre la calidad del software, y de la prueba, es aconsejable contar con criterios que definan cuándo debe comenzar una actividad de prueba dada y cuándo la actividad está completa. Los criterios de entrada (llamados, habitualmente, definición de preparado en desarrollo Ágil) definen las precondiciones para emprender una actividad de prueba específica. Si no se cumplen los criterios de entrada, es probable que la actividad resulte más difícil, más lenta, más costosa y más arriesgada. Los criterios de salida (más habitualmente llamados definición de hecho en el desarrollo Ágil) definen qué condiciones deben ser alcanzadas para poder afirmar que un nivel de prueba o un conjunto

[59] **Consultar ****

de pruebas han sido completados. Se deberían definir criterios de entrada y salida para cada nivel de prueba y tipo de prueba, y diferirán en función de los objetivos de la prueba.

Entre los criterios de entrada habituales se encuentran:

- Disponibilidad de requisitos, historias de usuarios y/o modelos (por ejemplo, cuando se sigue una estrategia de prueba basada en modelos) con capacidad de ser probados.
- Disponibilidad de elementos de prueba que han cumplido los criterios de salida para cualquiera de los niveles de prueba anteriores.
- Disponibilidad del entorno de prueba.
- Disponibilidad de las herramientas de prueba necesarias.
- Disponibilidad de datos de prueba y otros recursos necesarios.

Entre los criterios de salida habituales se encuentran:

- Las pruebas planificadas han sido ejecutadas.
- Se ha logrado un nivel de cobertura definido (por ejemplo, de requisitos, historias de usuario, criterios de aceptación, riesgos, código).
- El número de defectos no resueltos se encuentra dentro de un límite acordado.
- El número estimado de defectos restantes es suficientemente bajo.
- Los niveles de fiabilidad, eficiencia de desempeño, usabilidad, seguridad y otras características de calidad relevantes son suficientes.

Incluso sin que se cumplan los criterios de salida, también es común que las actividades de prueba se reduzcan debido al presupuesto que se haya consumido, al tiempo previsto que se haya consumido, y/o a la presión para sacar el producto al mercado. Puede ser aceptable poner fin a las pruebas en tales circunstancias, si los implicados del proyecto y los propietarios del negocio han revisado y aceptado el riesgo de ponerlo en marcha sin más pruebas.

5.2.4 Calendario de Ejecución de Prueba

Una vez que los diversos casos de prueba y procedimientos de prueba han sido desarrollados (con algunos procedimientos de prueba potencialmente automatizados) y agrupados en juegos de prueba, los juegos de prueba pueden organizarse en un calendario de ejecución de prueba que define el orden en el que deben ejecutarse. El calendario de ejecución de la prueba debe tener en cuenta factores tales como la priorización, las dependencias, las pruebas de confirmación, las pruebas de regresión y la secuencia más eficaz para ejecutar las pruebas.

En condiciones ideales, los casos de prueba se ordenarían en función de sus niveles de prioridad, generalmente ejecutando primero los casos de prueba con mayor prioridad. Sin embargo, esta práctica puede no funcionar si los casos de prueba tienen dependencias o si las características 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, se debe ejecutar primero el caso de prueba con una prioridad más baja. Del mismo modo, si hay dependencias entre los casos de prueba, deben ordenarse adecuadamente, independientemente de sus prioridades relativas. Las pruebas de confirmación y regresión también deben estar priorizadas, basándose en la importancia de la retroalimentación rápida sobre los cambios, pero en este caso también puede haber dependencias.

En algunos casos, son posibles varias secuencias de pruebas, con diferentes niveles de eficiencia asociados a esas secuencias. En tales casos, debe haber compromisos entre la eficiencia de la ejecución de la prueba y el cumplimiento de las prioridades.

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 la prueba para un proyecto, un lanzamiento o una iteración en particular. Los factores que influyen en el esfuerzo de la prueba pueden incluir características del producto, características del proceso de desarrollo, 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 (por ejemplo, seguridad, fiabilidad).
- El nivel de detalle necesario para la documentación de la prueba.
- Requisitos para el cumplimiento de normas legales y reglamentarias.

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.
- Presión con respecto al tiempo.

Características de las Personas

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

Resultados de la Prueba

- El número y la severidad de los defectos detectados.
- La cantidad de reconstrucción[60] necesaria.

5.2.6 Técnicas de Estimación de la Prueba

Hay una serie de técnicas de estimación utilizadas para determinar el esfuerzo necesario para realizar la prueba de forma adecuada. Dos de las técnicas más utilizadas son:

- La técnica basada en métricas: estimación del esfuerzo de prueba basada en métricas de proyectos similares anteriores o en valores típicos.
- La técnica basada en expertos: estimar el esfuerzo de la prueba basándose en la experiencia de los propietarios de las tareas de prueba o por expertos.

Por ejemplo, en el desarrollo Ágil, los gráficos de trabajo pendiente son ejemplos del enfoque basado en métricas dado que el esfuerzo que es capturado e informado, luego utilizado para alimentar la velocidad del equipo a fin de determinar la cantidad de trabajo que el equipo puede hacer en la siguiente iteración; mientras que el póquer de planificación es un ejemplo de la aproximación basada en expertos, dado que los miembros del equipo estiman el esfuerzo para entregar una prestación en base a su experiencia (Programa de Estudio de Probador Certificado del ISTQB - Nivel Básico - Extensión Ágil - 2014).

En los proyectos secuenciales, los modelos de eliminación de defectos son ejemplos del enfoque basado en métricas, en el que se capturan y comunican los volúmenes de defectos y el tiempo necesario para eliminarlos, lo que proporciona una base para estimar futuros proyectos de naturaleza similar; mientras que la técnica de estimación Delphi de Banda Ancha es un ejemplo de la técnica de estimación basada en expertos en la que los grupos de expertos realizan estimaciones basadas en su experiencia (Programa de Estudio de Probador Certificado del ISTQB - Nivel Avanzado - Jefe de Prueba - 2012).

5.3 Monitorización y Control de la Prueba

El propósito de la monitorización de la prueba es reunir información y proporcionar retroalimentación y visibilidad sobre las actividades de prueba. La información que se debe monitorizar puede recopilarse 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, como por ejemplo, el cumplimiento de los objetivos en cuanto a la cobertura de los riesgos de producto, de los requisitos, o de los criterios de aceptación.

El control de la prueba describe cualquier acción de orientación[61] o 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 a cualquier otra actividad del ciclo de vida del software.

Entre los ejemplos de acciones de control de la prueba se incluyen:

- Volver a priorizar las pruebas cuando se produce un riesgo identificado (por ejemplo, cuando el software se entrega tarde).
- Cambiar el calendario de prueba debido a la disponibilidad o no disponibilidad de un entorno de prueba u otros recursos.

[60] **Consultar ****
[61] **Consultar ****

- Revaluar[62] si un elemento de prueba cumple un criterio de entrada o salida debido a un trabajo de reconstrucción.

5.3.1 Métricas Utilizadas en la Prueba

Las métricas se pueden recopilar durante y al final de las actividades de prueba con el fin de evaluar:

- Avance con respecto al calendario y presupuesto previstos.
- Calidad actual del objeto de prueba.
- Adecuación del enfoque de prueba.
- Eficacia de las actividades de prueba con respecto a los objetivos.

Entre las métricas de prueba comunes se encuentran:

- Porcentaje de trabajo planificado realizado en la preparación de los casos de prueba (o porcentaje de casos de prueba planificados implementados).
- Porcentaje de trabajo planificado realizado en la preparación del entorno de prueba.
- Ejecución de casos de prueba (por ejemplo, número de casos de prueba ejecutados/no ejecutados, casos de prueba pasados/fallados, y/o condiciones de prueba pasadas/falladas).
- Información sobre defectos (por ejemplo, densidad de defectos, defectos encontrados y corregidos, tasa de fallos y resultados de la prueba de confirmación).
- Cobertura de prueba con respecto a requisitos, historias de usuarios, criterios de aceptación, riesgos o código.
- La finalización de una tarea, la asignación y uso de recursos y el esfuerzo.
- El coste de la prueba, incluyendo el coste comparado con el beneficio de encontrar el próximo defecto o el coste comparado con el beneficio de realizar la siguiente prueba.

5.3.2 Objetivos, Contenidos y Audiencias de los Informes de Prueba

El propósito del informe de prueba es resumir y comunicar la información de la actividad de prueba, tanto durante como al final de una actividad de prueba (por ejemplo, un nivel de prueba). El informe de prueba elaborado durante una actividad de prueba puede denominarse informe de avance de la prueba, mientras que un informe de prueba preparado al final de una actividad de prueba puede denominarse informe resumen de la prueba.

Durante la monitorización y el control de la prueba, el jefe de la prueba emite periódicamente informes de avance de la prueba para los implicados. Además del contenido común de los informes de avance de la prueba y de los informes de resumen de la prueba, los informes de avance de la prueba típicos también pueden incluir:

- El estado de las actividades de prueba y el avance con respecto al plan de prueba.
- Factores que impiden el avance.
- Pruebas previstas para el próximo período objeto de informe.
- La calidad del objeto de prueba.

[62] **Consultar ****

Cuando se alcanzan los criterios de salida, el jefe de la prueba emite el informe de resumen de la prueba. Este informe proporciona un resumen de la prueba realizada, basado en el último informe de avance de la prueba y cualquier otra información relevante.

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

- Resumen de la prueba realizada.
- Información sobre lo ocurrido durante un período de prueba.
- Desviaciones con respecto al plan, incluidas las desviaciones en el calendario, la duración o el esfuerzo de las actividades de prueba.
- Estado de la 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 la prueba, avance de la actividad y consumo de recursos. (por ejemplo, como se describe en el punto 5.3.1)
- Riesgos residuales (véase el apartado 5.5).
- Productos de trabajo de la prueba reutilizables desarrollados.

El contenido de un informe de prueba variará dependiendo del proyecto, los requisitos de la organización y el ciclo de vida de desarrollo de software. Por ejemplo, un proyecto complejo con muchos implicados o un proyecto regulado pueden requerir informes más detallados y rigurosos que una actualización de software rápida. Como otro ejemplo, en el desarrollo Ágil, el informe de avance de la prueba puede estar incorporado en los tableros de tareas[63], resúmenes de defectos y gráficos de trabajo pendiente, los cuales pueden ser discutidos durante una reunión de pie[64] diaria (ver Programa de Estudio de Probador Certificado del ISTQB - Nivel Básico - Extensión Ágil - 2014).

Además de adaptar los informes de prueba en función del 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 pueden ser diferentes de las que se incluiría en un informe de resumen ejecutivo. En el primer caso, la información detallada sobre los tipos de defectos y las tendencias pueden ser importantes. En este último caso, un informe de alto nivel (por ejemplo, un resumen de la situación de los defectos por prioridad, presupuesto, calendario y condiciones de prueba pasadas/falladas/no probadas) puede ser más adecuado.

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

[63] **Consultar ****
[64] **Consultar ****

5.4 Gestión de la Configuración

El objetivo de la gestión de configuración es establecer y mantener la integridad del componente o sistema, los productos de prueba y sus relaciones entre sí a lo largo del ciclo de vida del proyecto y del producto.

Para dar un soporte adecuado a la prueba, 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 hace un seguimiento de los cambios y se relacionan entre sí.
- Todos los elementos correspondientes a productos de prueba se identifican de forma única, se controlan las versiones, se hace un seguimiento de los cambios, se relacionan entre sí y se relacionan con las versiones de los elementos de prueba, de modo que se pueda mantener la trazabilidad durante todo el proceso de prueba.
- Todos los documentos y elementos software identificados están referenciados de forma inequívoca[65] en la documentación de la prueba.

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

5.5 Riesgos y Prueba

5.5.1 Definición de Riesgo

El riesgo implica la posibilidad de que ocurra 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[66]) de ese evento.

5.5.2 Riesgos de Producto y Riesgos de Proyecto

El riesgo de producto implica la posibilidad de que un producto de trabajo (por ejemplo, una especificación, componente, sistema o prueba) pueda no satisfacer las necesidades legítimas de sus usuarios y/o implicados. Cuando los riesgos de producto están asociados a características de calidad específicas de un producto (por ejemplo, adecuación funcional, fiabilidad, eficiencia de desempeño, usabilidad, seguridad, compatibilidad, mantenibilidad y portabilidad), los riesgos de producto también se denominan riesgos de la calidad. Entre los ejemplos de riesgos de producto se encuentran:

- El software podría no realizar las funciones previstas de acuerdo con la especificación.
- El software puede no realizar las funciones previstas de acuerdo con las necesidades de los usuarios, clientes y/o implicados.
- Una arquitectura de sistema puede no soportar de forma adecuada algunos requisitos no funcionales.
- Un cómputo específico puede realizarse de forma incorrecta en algunas circunstancias.
- Una estructura de control de un bucle puede estar codificada de forma incorrecta.

[65] **Consultar ****
[66] **Consultar ****

- Los tiempos de respuesta pueden ser inadecuados para un sistema de procesamiento de transacciones de alto rendimiento.
- La retroalimentación de la experiencia de usuario (UX[67] por sus siglas en inglés) podría no cumplir con las expectativas respecto del producto.

El riesgo de proyecto implica situaciones que, en caso de que ocurrieran, podrían tener un efecto negativo en la capacidad de un proyecto para lograr sus objetivos. Entre los ejemplos de riesgos del proyecto se incluyen:

- Cuestiones Asociadas al Proyecto:
  - Se pueden producir retrasos en la entrega, la finalización de tareas, el cumplimiento de los criterios de salida o la definición de hecho.
  - Las estimaciones inexactas, la reasignación de fondos a proyectos de mayor prioridad o el recorte general de gastos en toda la organización pueden dar lugar a una financiación inadecuada.
  - Los cambios tardíos pueden resultar en cambios sustanciales de reelaboración[68].
- Cuestiones Asociadas a la Organización:
  - Las competencias, la formación y el personal pueden no ser suficientes.
  - Las cuestiones relacionadas con el personal pueden causar conflictos y problemas.
  - Los usuarios, el personal de la empresa o los expertos en la materia pueden no estar disponibles debido a prioridades de negocio en conflicto.
- Cuestiones de Carácter Político:
  - Los probadores pueden no comunicar adecuadamente sus necesidades y/o los resultados de la prueba.
  - Los desarrolladores y/o probadores pueden no realizar un seguimiento de la información obtenida en la prueba y las revisiones (por ejemplo, no mejorar las prácticas de desarrollo y prueba).
  - Puede haber una actitud y expectativas inadecuadas con respecto a la prueba (por ejemplo, no apreciar el valor de encontrar defectos durante la prueba).
- Cuestiones de Carácter Técnico:
  - Es posible que los requisitos no estén suficientemente bien definidos.
  - Es posible que no se cumplan los requisitos, dadas las restricciones existentes.
  - Es posible que el entorno de prueba no esté listo a tiempo.
  - La conversión de datos, la planificación de la migración y el soporte de herramientas se pueden retrasar.
  - 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.

[67] **Consultar ****
[68] **Consultar ****

- Una mala gestión de los defectos y problemas similares pueden dar lugar a defectos acumulados y otra deuda técnica[69].

- Cuestiones Relacionados con los Proveedores:
  - Un tercero puede no entregar un producto o servicio necesario, o declararse en quiebra.
  - Las cuestiones contractuales pueden causar problemas al proyecto.

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

5.5.3 La Prueba Basada en el Riesgo y la Calidad de Producto

El riesgo se utiliza para distribuir el esfuerzo necesario durante la prueba. Se utiliza para decidir dónde y cuándo empezar a realizar la prueba y para identificar las áreas que necesitan más atención. La prueba se utiliza para reducir la probabilidad de que ocurra un evento adverso o para reducir el impacto de un evento adverso. La prueba se utiliza como una actividad de mitigación del riesgo, para proporcionar retroalimentación sobre los riesgos identificados, así como para proporcionar retroalimentación sobre los riesgos residuales (no resueltos).

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

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

- Determinar las técnicas de prueba que se van a utilizar.
- Determinar los niveles y tipos particulares de pruebas que se realizarán (por ejemplo, pruebas de seguridad, pruebas de accesibilidad).
- Determinar el alcance de la prueba que se llevará a cabo.
- Priorizar la prueba en un intento de encontrar los defectos críticos tan pronto como sea posible.
- Determinar si se podrían realizar otras actividades además de las pruebas para reducir el riesgo (por ejemplo, impartir formación a diseñadores sin experiencia).

La prueba basada en el riesgo se basa en el conocimiento colectivo y la percepción de los implicados en el proyecto para llevar a cabo el análisis de riesgo de producto. Para asegurar que la probabilidad de que un producto falle se reduzca al mínimo, las actividades de gestión de riesgos aportan un enfoque disciplinado[70]:

- Analizar (y reevaluar regularmente) lo que puede salir mal (riesgos).
- Determinar qué riesgos son los que es importante tratar.
- Implementar acciones para mitigar esos riesgos.

[69] **Consultar ****
[70] **Consultar ****

- Elaborar planes de contingencia para hacer frente a los riesgos en caso de que se conviertan en hechos reales.

Además, la prueba puede identificar nuevos riesgos, ayudar a determinar qué riesgos deben ser mitigados y reducir la incertidumbre sobre los riesgos.

5.6 Gestión de Defectos

Dado que uno de los objetivos de la prueba es detectar defectos, los defectos encontrados durante la prueba deben registrarse. La forma en que se registran los defectos puede variar, dependiendo del contexto del componente o sistema que se está probando, el nivel de prueba y el modelo de ciclo de vida de desarrollo de software. Cualquier defecto identificado debe ser investigado y debe ser objeto de seguimiento[71] desde su descubrimiento y clasificación hasta su resolución (por ejemplo, la corrección de los defectos y las pruebas de confirmación de la solución, el aplazamiento a una liberación posterior, la aceptación como limitación permanente del producto, etc.). Para gestionar todos los defectos hasta su resolución, la organización debe establecer un proceso de gestión de defectos que incluya un flujo de trabajo y reglas de clasificación. Este proceso debe ser acordado con todos aquellos que participan en la gestión de defectos, incluyendo diseñadores, desarrolladores, probadores y propietarios de producto. 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 llegar a describir falsos positivos, no fallos reales debido a defectos. Por ejemplo, una prueba puede fallar cuando se interrumpe una conexión de red o cuando se agota el tiempo de espera. Este comportamiento no resulta de un defecto en el objeto de prueba, sino que es una anomalía que debe ser investigada. Los probadores deben tratar de minimizar el número de falsos positivos informados como defectos.

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

Los informes de defecto típicos tienen los siguientes objetivos:

- Proporcionar a los desarrolladores y otras partes información sobre cualquier evento adverso que haya ocurrido, para que puedan identificar efectos específicos, aislar el problema con una prueba de reproducción mínima y corregir los defectos potenciales, según sea necesario, o resolver el problema de otra manera.
- Proporcionar a los jefes de prueba un medio para hacer un seguimiento de la calidad del producto de trabajo y del impacto en la prueba (por ejemplo, si se notifican muchos defectos, los encargados de la prueba habrán dedicado mucho tiempo a informarlos en lugar de realizar pruebas, y será necesario realizar más pruebas de confirmación).
- Proporcionar ideas para la mejora de los procesos de desarrollo y prueba.

[71] **Consultar ****

Un informe de defecto archivado durante una prueba dinámica habitualmente incluye:

- Un identificador.
- Un título y un breve resumen del defecto que se está informando.
- Fecha del informe de defecto, organización emisora y autor.
- Identificación del elemento de prueba (elemento de configuración que se está probando) y del entorno.
- La(s) fase(s) del ciclo de vida de desarrollo en la(s) que se observó el defecto.
- Una descripción del defecto para permitir la reproducción y resolución, incluyendo registros, capturas de pantalla de volcados de bases de datos o grabaciones (si se detectan durante la ejecución de la prueba).
- Resultados esperados y reales.
- Alcance o grado de impacto (severidad) del defecto con respecto a los intereses de los implicados.
- Urgencia/prioridad de la corrección.
- Estado del informe de defecto (por ejemplo, abierto, diferido, duplicado, a la espera de ser corregido, a la espera de la prueba de confirmación, reabierto, cerrado).
- Conclusiones, recomendaciones y aprobaciones.
- Cuestiones generales, tales como otras áreas que pueden verse afectadas por un cambio resultante del defecto.
- Historial de cambios, como la secuencia de acciones tomadas por los miembros del equipo de proyecto con respecto al defecto para aislarlo, repararlo y confirmar que ha sido corregido.
- Referencias, incluyendo el caso de prueba que puso en evidencia el problema.

Algunos de estos detalles pueden incluirse y/o gestionarse automáticamente cuando se utilizan herramientas de gestión de defectos, por ejemplo, la asignación automática de un identificador, la asignación y actualización del estado del informe de defecto durante el flujo de trabajo, etc. Los defectos encontrados durante las pruebas estáticas, particularmente las revisiones, normalmente se documentarán de una manera diferente, por ejemplo, en las notas de las reuniones de revisión.

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

6 Soporte de Herramientas para el Proceso de Prueba

Duración: 40 minutos

Palabras Clave

prueba guiada por datos ("data-driven testing")
prueba guiada por palabra clave ("keyword-driven testing")
herramienta de prueba de rendimiento ("performance testing tool")
automatización de la prueba ("test automation")
herramienta de ejecución de la prueba ("test execution tool")
herramienta de gestión de la prueba ("test management tool")

Objetivos de Aprendizaje para Herramientas de Prueba

6.1 Consideraciones sobre las Herramientas de Prueba

NB-6.1.1 (K2) Clasificar las herramientas de prueba de acuerdo con su finalidad y las actividades de prueba a las que dan soporte.

NB-6.1.2 (K1) Identificar los beneficios y riesgos de la automatización de la prueba.

NB-6.1.3 (K1) Recordar consideraciones especiales para las herramientas de ejecución de pruebas y las herramientas gestión de pruebas.

6.2 Uso Eficaz de las Herramientas

NB-6.2.1 (K1) Identificar los principios básicos para seleccionar una herramienta.

NB-6.2.2 (K1) Recordar los objetivos de utilizar proyectos piloto para introducir herramientas.

NB-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 utilizar para apoyar una o más actividades de prueba. Entre estas herramientas se encuentran:

- Herramientas que se utilizan directamente en la prueba, como herramientas de ejecución de la prueba y herramientas de preparación de datos de prueba.
- Herramientas que ayudan a gestionar requisitos, casos de prueba, procedimientos de prueba, guiones de prueba automatizados, resultados de prueba, datos de prueba y defectos, así como a informar y monitorizar la ejecución de la prueba.
- Herramientas que se utilizan para la investigación y la evaluación.
- Cualquier herramienta que asista en la prueba (una hoja de cálculo es también 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 objetivos, dependiendo del contexto:

- Mejorar la eficiencia de las actividades de prueba mediante la automatización de tareas repetitivas o que requieren recursos significativos cuando se realizan manualmente (por ejemplo, ejecución de pruebas, prueba de regresión).
- Mejorar la eficiencia de las actividades de prueba apoyando las actividades de prueba manuales a lo largo de todo el proceso de prueba (véase 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 (por ejemplo, pruebas de rendimiento a gran escala).
- Aumentar la fiabilidad de las pruebas (por ejemplo, automatizando las comparaciones de grandes cantidades de datos o simulando el comportamiento).

Las herramientas de prueba pueden clasificarse en función de varios criterios, como el objetivo, el precio, el modelo de concesión de licencias (por ejemplo, comercial o de código abierto) y la tecnología utilizada. En este programa de estudio las herramientas de prueba se clasifican según las actividades de prueba a las que dan soporte.

Algunas herramientas claramente dan soporte a una sola o principalmente a una actividad; otras pueden dar soporte a 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 ser suministradas como un paquete integrado.

Algunos tipos de herramientas de prueba pueden ser intrusivas, lo que significa que pueden afectar al resultado real de la prueba. Por ejemplo, los tiempos de respuesta reales de una aplicación pueden ser diferentes debido a las instrucciones adicionales que son ejecutadas por una herramienta de pruebas de rendimiento, o la cantidad de cobertura de código alcanzada puede estar distorsionada debido al uso de una herramienta de cobertura. La consecuencia de utilizar herramientas intrusivas se conoce como efecto sonda[72].

Algunas herramientas ofrecen soporte que normalmente es más adecuado para los desarrolladores (por ejemplo, herramientas que se utilizan durante la prueba de componentes e integración). Estas herramientas se identifican con "(D)" en las secciones siguientes.

Soporte de Herramientas para la Gestión de la Prueba y Productos de Prueba

Las herramientas de gestión se pueden utilizar en cualquier actividad de prueba a lo largo de todo el ciclo de vida de desarrollo de software. Algunos ejemplos de herramientas que dan soporte a la gestión de la prueba y productos de prueba incluyen:

- Herramientas de gestión de prueba y herramientas de gestión del ciclo de vida[73] de las aplicaciones (ALM por sus siglas en inglés).
- Herramientas de gestión de requisitos (por ejemplo, trazabilidad de los objetos de prueba).
- Herramientas de gestión de defectos.
- Herramientas de gestión de la configuración.
- Herramientas de integración continua (D).

[72] **Consultar ****
[73] **Consultar ****

Soporte de Herramientas para la Prueba Estática

Las herramientas para la prueba estática están asociadas a las actividades y beneficios descritos en el capítulo 3. Algunos ejemplos de estas herramientas incluyen:

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

Soporte de Herramientas para el Diseño e Implementación de Pruebas

Las herramientas de diseño de pruebas ayudan a crear productos de trabajo mantenibles en el diseño e implementación de pruebas, incluyendo casos de prueba, procedimientos de prueba y datos de prueba. Algunos ejemplos de estas herramientas incluyen:

- Herramientas de diseño de pruebas.
- Herramientas de Prueba Basada en Modelos.
- Herramientas de preparación de datos de prueba.
- Herramientas de desarrollo guiado por prueba de aceptación[74] (ATDD por sus siglas en inglés) y de desarrollo guiado por el comportamiento[75] (BDD por sus siglas en inglés).
- Herramientas de desarrollo guiado por pruebas (DGP)[76] (D).

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

Soporte de Herramientas para la Ejecución y el Registro de Pruebas

Hay muchas herramientas para apoyar y mejorar las actividades de ejecución y registro de pruebas. Algunos ejemplos de estas herramientas incluyen:

- Herramientas de ejecución de pruebas (por ejemplo, para ejecutar pruebas de regresión).
- Herramientas de cobertura (por ejemplo, cobertura de requisitos, cobertura de código (D)).
- Arneses de prueba (D).
- Herramientas de marco de trabajo de pruebas unitarias (D).

Soporte de Herramientas para la Medición del Rendimiento y el Análisis Dinámico

Las herramientas para la medición del rendimiento y el análisis dinámico son esenciales para dar soporte a las actividades de pruebas de rendimiento y de carga, ya que estas actividades no se pueden realizar de forma eficaz de forma manual. Algunos ejemplos de estas herramientas incluyen:

- Herramientas para la prueba de rendimiento.
- Herramientas de monitorización.
- Herramientas de análisis dinámico (D).

[74] **Consultar ****
[75] **Consultar ****
[76] **Consultar ****

Soporte de Herramientas para Necesidades de Prueba Especializadas

Además de las herramientas que soportan el proceso general de prueba, hay muchas otras herramientas que dan soporte a cuestiones más específicas de la prueba. Ejemplos de esto incluyen herramientas que se centran en:

- Evaluación de la calidad de los datos.
- Conversión y migración de datos.
- Prueba de usabilidad.
- Prueba de accesibilidad.
- Prueba de localización.
- Prueba de seguridad.
- Prueba de portabilidad (por ejemplo, la prueba del software en varias plataformas compatibles).

6.1.2 Beneficios y Riesgos de la Automatización de la Prueba

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 la prueba, pero también existen riesgos. Esto es particularmente cierto en el caso de las herramientas de ejecución de pruebas (lo que a menudo se denomina automatización de la prueba).

Los beneficios potenciales del uso de herramientas para apoyar la ejecución de la prueba incluyen:

- Reducción del trabajo manual repetitivo (por ejemplo, la ejecución de pruebas de regresión, las tareas de configuración y desmontaje del entorno, la reintroducción de los mismos datos de prueba y la comparación con los estándares de codificación), lo que ahorra tiempo.
- Mayor consistencia y repetibilidad (por ejemplo, los datos de las pruebas se crean de manera coherente, las pruebas se ejecutan con una herramienta en el mismo orden y con la misma frecuencia, y las pruebas se generan de forma consistente a partir de los requisitos).
- Una evaluación más objetiva (por ejemplo, mediciones estáticas, cobertura).
- Acceso más fácil a la información sobre las pruebas (por ejemplo, estadísticas y gráficos sobre el avance de la prueba, las tasas de defectos y el rendimiento).

Los riesgos potenciales de utilizar herramientas para dar soporte a la prueba incluyen:

- Las expectativas con respecto a la herramienta pueden ser poco realistas (incluyendo la funcionalidad y la facilidad de uso).
- El tiempo, coste y esfuerzo para la introducción inicial de una herramienta pueden haber sido subestimados (incluyendo la capacitación y la experiencia externa).
- El tiempo y el esfuerzo necesarios para lograr beneficios significativos y continuos de la herramienta pueden haber sido subestimados (incluyendo 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 activos de prueba generados por la herramienta puede haber sido 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 la prueba manual sería mejor).

- El control de versiones de los activos de prueba puede ser descuidado.
- Las relaciones y las cuestiones de interoperabilidad entre las herramientas críticas pueden descuidarse, como las herramientas de gestión de requisitos, las herramientas de gestión de configuración, las herramientas de gestión de defectos y las herramientas de distintos proveedores.
- El proveedor de herramientas puede cerrar el negocio, retirar la herramienta o venderla a otro proveedor.
- El proveedor puede proporcionar una respuesta deficiente para soporte, actualizaciones y correcciones de defectos.
- Un proyecto de código abierto[77] puede ser suspendido.
- Una nueva plataforma o tecnología puede no estar soportada por la herramienta.
- Es posible que no haya una titularidad clara de la herramienta (por ejemplo, para tutorías, actualizaciones, etc.).

6.1.3 Consideraciones Especiales con Respecto a las Herramientas de Ejecución y Gestión de Prueba

Para que la implementación sea fluida y exitosa, hay una serie de cosas que se deben tener en cuenta al seleccionar e integrar herramientas de ejecución de pruebas y de gestión de pruebas en una organización.

Herramientas de Ejecución de Prueba

Las herramientas de ejecución de prueba 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 escala a un gran número de secuencias de guiones de prueba. Un guion capturado[78] es una representación lineal con datos y acciones específicas como parte de cada guion. Este tipo de guion puede ser inestable cuando ocurren eventos inesperados. La última generación de estas herramientas, que aprovecha la tecnología de captura "inteligente" de imágenes, 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 prueba guiada por datos separa las entradas y los resultados esperados de la prueba, generalmente en una hoja de cálculo, y utiliza un guion de prueba más genérico que puede leer los datos de entrada y ejecutar el mismo guion de prueba con datos diferentes. Los probadores que no están familiarizados con el lenguaje de guion pueden crear nuevos datos de prueba para estos guiones predefinidos.

En un enfoque de prueba guiada por palabra clave, un guion genérico procesa las palabras clave que describen las acciones que se deben llevar a cabo (también llamadas palabras de acción), que luego llama a los guiones de palabra clave para procesar los datos de prueba asociados. Los probadores (incluso si no están familiarizados con el lenguaje de guion) pueden definir pruebas utilizando las palabras clave y los datos asociados, que pueden adaptarse a la aplicación que se está probando. Más detalles y ejemplos de enfoques de prueba guiada por datos y palabra clave se encuentran en el programa de estudio de Probador

[77] **Consultar ****
[78] **Consultar ****

Certificado del ISTQB - Nivel Avanzado - Ingeniero de Automatización de Pruebas - 2016, Fewster 1999 y Buwalda 2001.

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

Las herramientas de Prueba Basada en Modelos (PBM) permiten capturar una especificación funcional en forma de modelo, como un diagrama de actividad. En general, esta tarea la realiza un diseñador de sistemas. La herramienta de PBM interpreta el modelo para crear especificaciones de casos de prueba que pueden ser guardadas en una herramienta de gestión de pruebas y/o ejecutadas por una herramienta de ejecución de pruebas (ver el programa de estudio de Probador Certificado del ISTQB - Nivel Básico - Extensión Prueba Basada en Modelos - 2015).

Herramientas de Gestión de Prueba

Las herramientas de gestión de prueba a menudo necesitan interactuar con otras herramientas u hojas de cálculo por varias razones, incluyendo:

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

Esto es particularmente importante cuando se utiliza una herramienta integrada (por ejemplo, Gestión del Ciclo de Vida de la Aplicación[79]), que incluye un módulo de gestión de pruebas (y posiblemente un sistema de gestión de defectos), así como otros módulos (por ejemplo, información sobre el calendario y el presupuesto del proyecto) que son utilizados por diferentes grupos dentro de una organización.

6.2 Uso Eficaz de las Herramientas

6.2.1 Principios Básicos para la Selección de Herramientas

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

- Evaluación de la madurez de la organización, sus fortalezas y debilidades.
- Identificar oportunidades para un proceso de prueba mejorado con el soporte de herramientas.
- Comprender las tecnologías utilizadas por el objeto u objetos de prueba, con el fin de seleccionar una herramienta que sea compatible con dicha tecnología.
- Las herramientas de compilación e integración continua ya en uso dentro de la organización, con el fin de asegurar la compatibilidad e integración de las herramientas.
- Evaluar la herramienta con respecto a requisitos claros y criterios objetivos.

[79] **Consultar ****

- Tener en cuenta si la herramienta está disponible durante un período de prueba gratuito (y durante cuánto tiempo).
- Evaluar al proveedor (incluyendo formación, soporte y aspectos comerciales) o soporte para herramientas no comerciales (por ejemplo, de código abierto).
- Identificar los requisitos internos para el entrenamiento[80] y asesoramiento[81] en el uso de la herramienta.
- Evaluar las necesidades de formación, teniendo en cuenta las competencias en materia de prueba (y automatización de la prueba) de quienes trabajarán directamente con la(s) herramienta(s).
- Considerar los pros y contras de varios modelos de licenciamiento (por ejemplo, comercial o de código abierto).
- Estimar la relación coste-beneficio a partir de un caso de negocio concreto (si fuera necesario).

Como paso final, se debe realizar la evaluación de una prueba de concepto para establecer si la herramienta funciona eficazmente con el software sujeto a prueba y dentro de la infraestructura actual o, si fuera preciso, identificar los cambios necesarios en dicha infraestructura para utilizar la herramienta de manera eficaz.

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 positiva, 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, entendiendo tanto sus fortalezas como sus debilidades.
- Evaluar cómo encaja la herramienta con los procesos y prácticas existentes, y determinar qué se necesita cambiar.
- Decidir sobre las formas estándar de usar, administrar, almacenar y mantener la herramienta y los activos de prueba (por ejemplo, decidir sobre convenciones de nombres para archivos y pruebas, seleccionar estándares de codificación, crear bibliotecas y definir la modularidad de los juegos de prueba).
- Evaluar si los beneficios se lograrán a un coste razonable.
- Comprender las métricas que se desea que la herramienta recopile e informe, y configurar la herramienta para asegurar que estas métricas puedan ser capturadas e informadas.

6.2.3 Factores de Éxito para Herramientas

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

- Despliegue de la herramienta al resto de la organización de forma incremental.
- Adaptar y mejorar los procesos para que se ajusten al uso de la herramienta.

[80] **Consultar ****
[81] **Consultar ****

- Proporcionar formación, entrenamiento y asesoramiento para los usuarios de la herramienta.
- Definición de directrices para el uso de la herramienta (por ejemplo, normas internas para la automatización).
- Implementar una forma de recopilar información de uso a partir del uso real de la herramienta.
- Monitorizar el uso y los beneficios de la herramienta.
- Proporcionar apoyo a los usuarios de una herramienta determinada.
- Recopilar las lecciones aprendidas de todos los usuarios.

También es importante garantizar que la herramienta se integre técnica y desde una perspectiva de la organización en el ciclo de vida del desarrollo de software, lo que puede implicar la participación de organizaciones independientes responsables de las operaciones y/o de terceros proveedores.

Ver Graham 2012 para experiencias y consejos sobre el uso de herramientas de ejecución de pruebas.

7 Referencias

7.1 Estándares

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

7.2 Documentos del ISTQB

ISTQB Glossary.

ISTQB Foundation Level Overview 2018.

ISTQB-MBT Foundation Level Model-Based Tester Extension Syllabus.

ISTQB-AT Foundation Level Agile Tester Extension Syllabus.

ISTQB-ATA Advanced Level Test Analyst Syllabus.

ISTQB-ATM Advanced Level Test Manager Syllabus.

ISTQB-SEC Advanced Level Security Tester Syllabus.

ISTQB-TAE Advanced Level Test Automation Engineer Syllabus.

ISTQB-ETM Expert Level Test Management Syllabus.

ISTQB-EITP Expert Level Improving the Test Process Syllabus.

7.3 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

7.4 Otros recursos (no referenciados 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 - Antecedentes del Programa de Estudio

8.1 Historia de este Documento

Este documento es el Programa de Estudio de Nivel Probador Certificado del ISTQB de Nivel Básico, el primer nivel de cualificación internacional aprobado por el ISTQB (www.istqb.org).

Este documento fue preparado entre 2014 y 2018 por un Grupo de Trabajo compuesto por miembros designados por el International Software Testing Qualifications Board (ISTQB). La versión 2018 fue revisada inicialmente por representantes de todos los comités miembro del ISTQB, y luego por representantes de la comunidad internacional de prueba de software.

8.2 Objetivos de la Cualificación del Certificado Básico

- Obtener reconocimiento para la prueba como una especialidad esencial y profesional en ingeniería de software.
- Proporcionar un marco de referencia estándar para el desarrollo de las carreras profesionales de los probadores.
- Permitir que los probadores profesionalmente cualificados sean reconocidos por los empleadores, clientes y sus pares en el ámbito profesional, y elevar el perfil de los probadores.
- Promover prácticas de prueba consistentes y buenas prácticas en todas las disciplinas de ingeniería de software.
- Identificar temas relativos a la prueba que sean relevantes y de valor para la industria.
- Permitir a los proveedores de software contratar a probadores certificados y obtener así una ventaja comercial sobre sus competidores mediante la publicidad de su política de contratación de probadores.
- Proporcionar una oportunidad para que los probadores y aquellos con interés en la prueba adquieran una cualificación reconocida internacionalmente en la materia.

8.3 Objetivos de la Cualificación Internacional

- Ser capaz de equiparar el conocimiento en materia de prueba entre los diferentes países.
- Permitir a los probadores desplazarse más fácilmente a través de las fronteras de los países.
- Permitir que los proyectos multinacionales/internacionales tengan una comprensión común de las cuestiones en materia de prueba.
- Incrementar el número de probadores cualificados en todo el mundo.
- Tener más impacto/valor como iniciativa de ámbito internacional que desde cualquier enfoque específico de un país.
- Desarrollar un cuerpo internacional común de comprensión y conocimiento en materia de prueba a través del programa de estudio y la terminología, y aumentar el nivel de conocimiento sobre la prueba para todos los participantes.
- Promover la práctica de la prueba como profesión en más países.
- Permitir a los probadores obtener una cualificación reconocida en su lengua materna.

- Permitir el intercambio de conocimiento y recursos entre países.
- Proporcionar reconocimiento internacional a los probadores y esta cualificación debido a la participación desde múltiples países.

8.4 Requisitos de Acceso a esta Cualificación

El criterio de admisión para poder realizar el examen de Probador Certificado del ISTQB - Nivel Básico es que los candidatos estén interesados en la prueba de software. Sin embargo, es muy recomendable que los candidatos también:

- Tener, al menos, un mínimo de experiencia en desarrollo de software o en la prueba de software, por ejemplo, seis meses de experiencia como probador de sistemas o de aceptación de usuarios o como desarrollador de software.
- Asistir a un curso que haya sido acreditado por uno de los comités miembro - reconocidos por el ISTQB según los estándares del ISTQB.

8.5 Antecedentes e Historia del Certificado Básico en Prueba de Software

La certificación independiente de los probadores de software comenzó en el Reino Unido con la British Computer Society's Information Systems Examination Board (ISEB), cuando se creó un Comité de Prueba de Software en 1998 (www.bcs.org.uk/iseb). En 2002, la ASQF de Alemania comenzó a apoyar un programa alemán para la cualificación de probadores (www.asqf.de). Este programa de estudio se basa en los programas de estudios de ISEB y ASQF; incluye contenidos reorganizados, actualizados y adicionales, y el énfasis está dirigido a los temas que proporcionarán la ayuda más práctica a los probadores.

Un Certificado Básico de Prueba de Software existente (por ejemplo, de ISEB, ASQF o de un comité miembro reconocido por ISTQB) otorgado antes de la emisión de este Certificado Internacional, se considerará equivalente al Certificado Internacional. El Certificado Básico no expira y no necesita ser renovado. La fecha en la que se haya otorgado se detalla en el Certificado.

En cada país participante, los aspectos locales son controlados por un Comité de Prueba de Software - reconocido por ISTQB a nivel nacional o regional. Los deberes de los comités miembro son especificados por el ISTQB, pero se implementan en cada país. Entre los deberes de los comités nacionales se espera que se incluyan la acreditación de los proveedores de formación y el establecimiento de exámenes.

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

Los siguientes objetivos de aprendizaje se definen como válidos para este programa de estudio. Cada tema del programa de estudio se examinará de acuerdo con el objetivo de aprendizaje para el mismo.

9.1 Nivel 1: Recordar (K1)

El candidato reconocerá y recordará un término o concepto.

Palabras Clave

identificar ("identify")
recordar ("remember" o "recall")
recuperar ("retrieve")
reconocer("recognize")
conocer ("know")

Ejemplos:

Se puede reconocer la definición de " fallo " como:

- "La no entrega del servicio a un usuario final o a cualquier otro implicado." o
- "Desviación del componente o sistema con respecto a su entrega, servicio o resultado esperado."

9.2 Nivel 2: Comprender (K2)

El candidato puede seleccionar las razones o explicaciones para las afirmaciones relacionadas con el tema, y puede resumir, comparar, clasificar, categorizar y dar ejemplos para el concepto de prueba.

Palabras Clave:

resumir ("summarize")
generalizar ("generalize")
abstraer ("abstract")
clasificar ("classify")
comparar ("compare")
mapear ("map")
contrastar ("contrast")
ejemplificar ("exemplify")
interpretar ("interpret")
traducir("translate")
representar ("represent")
inferir ("infer")
concluir ("conclude")
categorizar ("categorize")
construir modelos ("construct models")

Ejemplos:

Puede explicar la razón por la cual el análisis y el diseño de la prueba deben realizarse tan pronto como sea posible:

- Para detectar defectos cuando son más baratos de eliminar.

- Para detectar primero los defectos más importantes.

Puede explicar las similitudes y diferencias entre la prueba de integración y la prueba de sistema:

1 Similitudes: los objetos de prueba tanto para la prueba de integración como para la prueba de sistema incluyen más de un componente, y tanto la prueba de integración como la prueba de sistema pueden incluir tipos de prueba no funcionales.

- Diferencias: la prueba de integración se concentra en las interfaces e interacciones, y la prueba de sistema se concentra en los aspectos de todo el sistema, como el procesamiento de extremo a extremo.

9.3 Nivel 3: Aplicar (K3)

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

Palabras Clave:

implementar ("implement")
ejecutar ("execute")
utilizar ("use")
seguir un procedimiento ("follow a procedure")
aplicar un procedimiento ("apply a procedure")

Ejemplos:

- Puede identificar valores frontera para particiones válidas e inválidas.
- Puede seleccionar casos de prueba de un determinado diagrama de transición de estado 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 Publicación

El Programa de Estudio de Probador Certificado del ISTQB de Nivel Básico, versión de 2018 es una actualización y reescritura importante de la versión de 2011. Por esta razón, no hay notas de la publicación detalladas por capítulo y sección. Sin embargo, aquí se ofrece un resumen de los principales cambios. Además, en un documento separado de notas de la publicación, el ISTQB proporciona trazabilidad entre los objetivos de aprendizaje en la versión 2011 del Programa de Estudio de Nivel Básico y los objetivos de aprendizaje en la versión 2018 del Programa de Estudio de Nivel Básico, mostrando los que se han añadido, actualizado o eliminado.

A principios de 2017, más de 550.000 personas en más de 100 países han realizado el examen de nivel 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 de Nivel Básico para poder aprobar el examen, esto hace que el Programa de Estudio de Nivel Básico sea el documento de prueba de software más leído hasta ahora.

Esta importante actualización se hace con respecto a este patrimonio y para mejorar el valor que el ISTQB ofrece a las, aproximadamente, 500.000 personas en la comunidad mundial de prueba.

En esta versión, todos los objetivos de aprendizaje han sido editados para hacerlos atómicos, y para crear una trazabilidad clara desde cada objetivo de aprendizaje hasta la(s) sección(es) de contenido (y preguntas de examen) que están relacionadas con ese objetivo de aprendizaje, y para tener una trazabilidad clara desde la(s) sección(es) de contenido (y preguntas de examen) hasta el objetivo de aprendizaje asociado. Además, las asignaciones de tiempo de los capítulos se han hecho más realistas que las de la versión 2011 del programa de estudio, utilizando una heurística probada y fórmulas utilizadas con otros programas de estudio del ISTQB, que se basan en un análisis de los objetivos de aprendizaje que se tratarán en cada capítulo.

Si bien este es un programa de estudio de Nivel Básico, que expresa las mejores prácticas y técnicas que han resistido la prueba del tiempo, se han hecho cambios para modernizar la presentación del material, especialmente en términos de métodos de desarrollo de software (por ejemplo, Scrum y despliegue continuo) y tecnologías (por ejemplo, la Interne9999t de las Cosas). Se han actualizado los estándares de referencia para hacerlos más recientes de la siguiente manera:

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

Además, dado que la cartera del ISTQB ha crecido de forma significativa en la última década, se han añadido extensas referencias cruzadas a material relacionado en otros programas de estudio del ISTQB, donde sea relevante, así como una cuidadosa revisión para alinearlo con todos los programas de estudio y con el Glosario del ISTQB. El objetivo es hacer que esta versión sea más fácil de leer, entender, aprender y traducir, centrándose en aumentar la utilidad práctica y el equilibrio entre el conocimiento y las competencias.

Para un análisis detallado de los cambios realizados en esta versión, consulte el documento ISTQB Certified Tester Foundation Level Overview 2018.