Probador Certificado de ISTQB®

Programa de Estudio

Nivel Básico

Versión ES V01.01

Traducción realizada por

Spanish Software Testing Qualifications Board

con el apoyo de

Hispanic America Software Testing Board

Traducción del Programa de Estudio

"ISTQB® Certified Tester - Foundation Level, Version V4.0"

Nota sobre Derechos de Propiedad Intelectual

Historial de Revisiones

Versión	Fecha	Observaciones
CTFL v4.0	21.04.2023	CTFL v4.0 – General release version
CTFL v3.1.1	01.07.2021	CTFL v3.1.1 – Copyright and logo update
CTFL v3.1	11.11.2019	CTFL v3.1 – Maintenance release with minor updates
ISTQB 2018	27.04.2018	CTFL v3.0 – Candidate general release version
ISTQB 2011	1.04.2011	CTFL Syllabus Maintenance Release
ISTQB 2010	30.03.2010	CTFL Syllabus Maintenance Release
ISTQB 2007	01.05.2007	CTFL Syllabus Maintenance Release
ISTQB 2005	01.07.2005	Certified Tester Foundation Level Syllabus v1.0
ASQF V2.2	07.2003	ASQF Syllabus Foundation Level Version v2.2 “Lehrplan Grundlagen des Software-testens“
ISEB V2.0	25.02.1999	ISEB Software Testing Foundation Syllabus v2.0

Tabla de Contenidos

Agradecimientos

Este documento ha sido hecho público formalmente por la Asamblea General del ISTQB® el 21 de abril de 2023.

Fue elaborado por un equipo del "ISTQB joint Foundation Level & Agile Working Groups": Laura Albert, Renzo Cerquozzi (vicepresidente), Wim Decoutere, Klaudia Dussa-Zieger, Chintaka Indikadahena, Arnika Hryszko, Martin Klonk, Kenji Onishi, Michaël Pilaeten (copresidente), Meile Posthuma, Gandhinee Rajkomar, Stuart Reid, Eric Riou du Cosquer (copresidente), Jean-François Riverin, Adam Roman, Lucjan Stapp, Stephanie Ulrich (vicepresidenta), Eshraka Zakaria.

El equipo agradece a Stuart Reid, Patricia McQuaid y Leanne Howard su revisión técnica y al equipo revisor y a los Comités Miembro sus sugerencias y aportaciones.

Las siguientes personas participaron en la revisión, aportando comentarios y votando este programa de estudio: Adam Roman, Adam Scierski, Ágota Horváth, Ainsley Rood, Ale Rebon Portillo, Alessandro Collino, Alexander Alexandrov, Amanda Logue, Ana Ochoa, André Baumann, André Verschelling, Andreas Spillner, Anna Miazek, Armin Born, Arnd Pehl, Arne Becher, Attila Gyúri, Attila Kovács, Beata Karpinska, Benjamin Timmermans, Blair Mo, Carsten Weise, Chinthaka Indikadahena, Chris Van Bael, Ciaran O'Leary, Claude Zhang, Cristina Sobrero, Dandan Zheng, Dani Almog, Daniel Säther, Daniel van der Zwan, Danilo Magli, Darvay Tamás Béla, Dawn Haynes, Dena Pauletti, Dénes Medzihradszky, Doris Dötzer, Dot Graham, Edward Weller, Erhardt Wunderlich, Eric Riou Du Cosquer, Florian Fieber, Fran O'Hara, François Vaillancourt, Frans Dijkman, Gabriele Haller, Gary Mogyorodi, Georg Sehl, Géza Bujdosó, Giancarlo Tomasig, Giorgio Pisani, Gustavo Márquez Sosa, Helmut Pichler, Hongbao Zhai, Horst Pohlmann, Ignacio Trejos, Ilia Kulakov, Ine Lutterman, Ingvar Nordström, Iosif Itkin, Jamie Mitchell, Jan Giesen, Jean-Francois Riverin, Joanna Kazun, Joanne Tremblay, Joëlle Genois, Johan Klintin, John Kurowski, Jörn Münzel, Judy McKay, Jürgen Beniermann, Karol Frühauf, Katalin Balla, Kevin Kooh, Klaudia Dussa- Zieger, Klaus Erlenbach, Klaus Olsen, Krisztián Miskó, Laura Albert, Liang Ren, Lijuan Wang, Lloyd Roden, Lucjan Stapp, Mahmoud Khalaili, Marek Majernik, Maria Clara Choucair, Mark Rutz, Markus Niehammer, Martin Klonk, Márton Siska, Matthew Gregg, Matthias Hamburg, Mattijs Kemmink, Maud Schlich, May Abu-Sbeit, Meile Posthuma, Mette Bruhn-Pedersen, Michal Tal, Michel Boies, Mike Smith, Miroslav Renda, Mohsen Ekssir, Monika Stocklein Olsen, Murian Song, Nicola De Rosa, Nikita Kalyani, Nishan Portoyan, Nitzan Goldenberg, Ole Chr. Hansen, Patricia McQuaid, Patricia Osorio, Paul Weymouth, Pawel Kwasik, Peter Zimmerer, Petr Neugebauer, Piet de Roo, Radoslaw Smilgin, Ralf Bongard, Ralf Reissing, Randall Rice, Rik Marselis, Rogier Ammerlaan, Sabine Gschwandtner, Sabine Uhde, Salinda Wickramasinghe, Salvatore Reale, Sammy Kolluru, Samuel Ouko, Stephanie Ulrich, Stuart Reid, Surabhi Bellani, Szilard Szell, Tamás Gergely, Tamás Horváth, Tatiana Sergeeva, Tauhida Parveen, Thaer Mustafa, Thomas Eisbrenner, Thomas Harms, Thomas Heller, Tobias Letzkus, Tomas Rosenqvist, Werner Lieblang, Yaron Tsubery, Zhenlei Zuo y Zsolt Hargitai.
"ISTQB Working Group Foundation Level" (Edición 2018): Klaus Olsen (presidente), Tauhida Parveen (vicepresidente), Rex Black (director de proyecto), Eshraka Zakaria, Debra Friedenberg, Ebbe Munk, Hans Schaefer, Judy McKay, Marie Walsh, Meile Posthuma, Mike Smith, Radoslaw Smilgin, Stephanie Ulrich, Steve Toms, Corne Kruger, Dani Almog, Eric Riou du Cosquer, Igal Levi, Johan Klintin, Kenji Onishi, Rashed Karim, Stevan Zivanovic, Sunny Kwon, Thomas Müller, Vipul Kocher, Yaron Tsubery y a todos los Comités Miembro por sus sugerencias.

"ISTQB Working Group Foundation Level" (Edición 2018): Klaus Olsen (presidente), Tauhida Parveen (vicepresidente), Rex Black (director de proyecto), Eshraka Zakaria, Debra Friedenberg, Ebbe Munk, Hans Schaefer, Judy McKay, Marie Walsh, Meile Posthuma, Mike Smith, Radoslaw Smilgin, Stephanie Ulrich, Steve Toms, Corne Kruger, Dani Almog, Eric Riou du Cosquer, Igal Levi, Johan Klintin, Kenji Onishi, Rashed Karim, Stevan Zivanovic, Sunny Kwon, Thomas Müller, Vipul Kocher, Yaron Tsubery y a todos los Comités Miembro por sus sugerencias.

"ISTQB Working Group Foundation Level" (Edición 2011): Thomas Müller (presidente), Debra Friedenberg. El equipo central agradece al equipo revisor (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquier Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal), y a todos los Comités Miembro por las sugerencias para la versión actual del programa de estudio.

"ISTQB Working Group Foundation Level" (Edición 2005): Thomas Müller (presidente), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson y Erik van Veenendaal. El equipo principal da las gracias al equipo revisor y a todos los Comités Miembro por sus sugerencias.

Notas de la Versión en Idioma Español

Spanish Software Testing Qualifications Board (SSTQB) ha llevado a cabo la traducción del Programa de Estudio de "ISTQB® Certified Tester, Foundation Level" versión 4.0. Este Programa de Estudio se denomina, en idioma español, "Probador Certificado de ISTQB® de Nivel Básico" versión 4.0.

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

Responsable de la traducción:	Gustavo Márquez Sosa (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, 25 de julio de 2023

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 prueba, ingenieros de prueba, consultores de prueba, 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 jefes de proyecto, responsables de calidad, propietarios de producto, 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.

0.3 Carrera Profesional para Probadores

El esquema ISTQB® proporciona apoyo a los profesionales de la prueba en todas las etapas de su carrera ofreciendo tanto amplitud como profundidad de conocimientos. Las personas que hayan obtenido la certificación de la Fundación ISTQB® también pueden estar interesadas en el nivel avanzado troncal (analista de prueba, analista de pruebas técnicas y jefe de prueba) y a partir de esa certificación el nivel experto (Gestión de la Prueba o Mejora del Proceso de Prueba). Cualquiera persona que busque desarrollar competencias en prácticas de prueba en un entorno Ágil podría considerar las certificaciones de Probador Técnico Ágil o Liderazgo de Prueba Ágil a Escala. La corriente de Especialista ofrece una inmersión profunda en áreas que tienen enfoques de prueba específicos y actividades de prueba (por ejemplo, en automatización de la prueba, prueba de IA, prueba basada en modelos, prueba de aplicaciones móviles), que están relacionadas con áreas de prueba específicas (por ejemplo, prueba de rendimiento, prueba de usabilidad, prueba de aceptación, prueba de seguridad), o que agrupan conocimientos de prueba para ciertos dominios de la industria (por ejemplo, automoción o juegos). Visite www.istqb.org para obtener la información más reciente sobre el programa de probadores certificados de ISTQB.

0.4 Resultados de Negocio

En esta sección se enumeran los 14 resultados de negocio que se esperan de una persona que haya obtenido la certificación de nivel básico.

Un probador certificado de nivel básico puede...

FL - BO - 01	Comprender qué es probar y por qué es beneficioso
FL - BO - 02	Comprender los conceptos fundamentales de la prueba de software
FL - BO - 03	Identificar el enfoque de prueba y las actividades que se deben implementar en función del contexto de prueba
FL - BO - 04	Evaluar y mejorar la calidad de la documentación
FL - BO - 05	Aumentar la efectividad y eficiencia de la prueba
FL - BO - 06	Alinear el proceso de prueba con el ciclo de vida de desarrollo de software
FL - BO - 07	Comprender los principios de gestión de la prueba
FL - BO - 08	Redactar y comunicar informes de defectos claros y comprensibles
FL - BO - 09	Comprender los factores que influyen en las prioridades y esfuerzos relacionados con la prueba
FL - BO - 10	Trabajar como parte de un equipo interfuncional
FL - BO - 11	Conocer los riesgos y beneficios relacionados con la automatización de la prueba
FL - BO - 12	Identificar las competencias esenciales necesarias para probar
FL - BO - 13	Comprender el impacto del riesgo en las pruebas
FL - BO - 14	Informar eficazmente sobre el avance y la calidad de la prueba

0.5 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 los capítulos 1-6 de este programa de estudio son evaluables a un nivel K1. Es decir, se puede pedir al candidato que reconozca o recuerde una palabra clave o un concepto mencionado en cualquiera de los seis capítulos. Los niveles específicos de los objetivos de aprendizaje se muestran al principio de cada capítulo y se clasifican de la siguiente manera:

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

En el Apéndice A 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.

0.6 El Examen de Certificación

El examen de certificación de nivel básico se basa en este programa de estudio. Las respuestas a las preguntas del examen pueden requerir el uso de material basado en más de una sección de este programa de estudio. Todas las secciones del programa de estudio son objeto de examen, excepto la Introducción y los Apéndices. Los estándares y libros se incluyen como referencias (capítulo 7), pero su contenido no es evaluable, más allá de lo que se resume en el propio programa de estudio a partir de dichos estándares y libros. Consulte el documento "Foundation Level Examination Structures and Rules".

0.7 Acreditación

Un Comité Miembro de ISTQB® puede acreditar a los proveedores de formación cuyo material de curso siga este programa de estudio. Los proveedores de formación deberán obtener las directrices de acreditación del Comité Miembro u organismo que realice 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. Las directrices de acreditación para este programa de estudio siguen las "Accreditation Guidelines" generales publicadas por el "Processes Management and Compliance Working Group".

0.8 Tratamiento de Estándares

Existen estándares a los que se hace referencia en el programa de estudio básico (por ejemplo, estándares IEEE o ISO). Estas referencias proporcionan un marco (como en las referencias a la norma ISO 25010 relativa a las características de calidad) o para proporcionar una fuente de información adicional si así lo desea el lector. Los documentos estándar no están pensados para ser objeto de examen. Consulte el capítulo 7 para obtener más información sobre los estándares.

0.9 Actualización

La industria del software cambia rápidamente. Para hacer frente a estos cambios y proporcionar a los implicados acceso a información relevante y actualizada, los grupos de trabajo del ISTQB han creado enlaces en la página web www.istqb.org, que hacen referencia a la documentación de apoyo y a los cambios en los estándares. Esta información no es objeto de evaluación bajo el Programa de Estudio de Nivel Básico.

0.10 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 didácticos generales que describen el propósito del Nivel Básico.
- Una lista de términos (palabras clave) que los alumnos deben ser capaces de recordar.
- Objetivos de aprendizaje para cada área de conocimiento, que describen los resultados cognitivos del aprendizaje que se deben alcanzar.
- Una descripción de los conceptos clave, incluyendo referencias a fuentes reconocidas.

El contenido del programa de estudio no es una descripción de todo el área de conocimiento de la prueba de software; refleja el nivel de detalle que debe cubrirse en los cursos de formación de nivel básico. Se concentra en conceptos y técnicas de prueba que pueden aplicarse a todos los proyectos de software independientemente del CVDS empleado.

0.11 Organización de este Programa de Estudio

Hay seis capítulos con contenido que puede ser objeto de examen. El encabezamiento de nivel superior de cada capítulo especifica el tiempo de formación del mismo. No se proporciona cronología por debajo del nivel del capítulo. Para los cursos de formación acreditados, el programa de estudio exige un mínimo de 1135 minutos (18 horas y 55 minutos) de formación, distribuidos en los seis capítulos de la siguiente manera:

- Capítulo 1: Fundamentos del Proceso de Prueba (180 minutos)
  - El alumno aprende los principios básicos relacionados con la prueba, las razones por las que es necesario probar y cuáles son los objetivos de la prueba.
  - El alumno comprende el proceso de prueba, las principales actividades de prueba y el producto de prueba.
  - El alumno comprende las competencias esenciales para probar.
- Capítulo 2: Prueba a lo Largo del Ciclo de Vida de Desarrollo de Software (130 minutos)
  - El alumno aprende cómo se incorpora la prueba a los diferentes enfoques de desarrollo.
  - El alumno aprende los conceptos de los enfoques de probar primero, así como DevOps.
  - El alumno aprende sobre los diferentes niveles de prueba, tipos de prueba y prueba de mantenimiento.
- Capítulo 3: Prueba Estática (80 minutos)
  - El alumno aprende los fundamentos de la prueba estática, el proceso de retroalimentación y revisión.
- Capítulo 4: Análisis y Diseño de la Prueba (390 minutos)
  - El alumno aprende a aplicar técnicas de caja negra, caja blanca y prueba basada en la experiencia para obtener casos de prueba a partir de diversos productos software.
  - El alumno aprende sobre el enfoque de prueba basado en la colaboración.
- Capítulo 5: Gestión de la Prueba (335 minutos)
  - El alumno aprende a planificar las pruebas en general y a estimar el esfuerzo de la prueba.
  - El alumno aprende cómo los riesgos pueden influir en el alcance de la prueba.
  - El alumno aprende a monitorizar y controlar las actividades de prueba.
  - El alumno aprende cómo la gestión de la configuración apoya la prueba.
  - El alumno aprende a informar de los defectos de forma clara y comprensible.
- Capítulo 6: Soporte de Herramientas para el Proceso de Prueba (20 minutos)
  - El alumno aprende a clasificar las herramientas y a comprender los riesgos y las ventajas de la automatización de la prueba.

1 Fundamentos del Proceso de Prueba - 180 minutos

Duración: 180 minutos

Palabras Clave[1]

Español	Inglés
cobertura	coverage
depurar	debugging
defecto	defect
error	error
fallo	failure
calidad	quality
aseguramiento de la calidad	quality assurance
causa raíz	root cause
análisis de prueba	test analysis
base de prueba	test basis
caso de prueba	test case
compleción de la prueba	test completion
condición de prueba	test condition
control de la prueba	test control
datos de prueba	test data
diseño de la prueba	test design
ejecución de prueba	test execution
implementación de prueba	test implementation
monitorización de la prueba	test monitoring
objeto de prueba	test object
objetivo de prueba	test objective
planificación de prueba	test planning
procedimiento de prueba	test procedure
resultado de prueba	test result
prueba	testing
producto de prueba	testware

[1] Las palabras clave se encuentran ordenadas por orden alfabético de los términos en inglés.

validación	validation
verificación	verification

Objetivos de Aprendizaje para "Capítulo 1"

1.1 ¿Qué es Probar?		
FL-1.1.1	(K1)	Identificar objetivos de prueba característicos.
FL-1.1.2	(K2)	Diferenciar entre probar y depurar.
1.2 ¿Por qué es Necesario Probar?		
FL-1.2.1	(K2)	Aportar ejemplos de por qué es necesario realizar pruebas.
FL-1.2.2	(K1)	Recordar la relación entre prueba y aseguramiento de la calidad.
FL-1.2.3	(K2)	Distinguir entre causa raíz, error, defecto y fallo.
1.3 Principios de la Prueba		
FL-1.3.1	(K2)	Explicar los siete principios de la prueba.
1.4 Actividades de Prueba, Productos de Prueba y Roles de Prueba		
FL-1.4.1	(K2)	Resumir las diferentes actividades y tareas de prueba.
FL-1.4.2	(K2)	Explicar el impacto del contexto en el proceso de prueba.
FL-1.4.3	(K2)	Diferenciar el producto de prueba que soporta las actividades de prueba.
FL-1.4.4	(K2)	Explicar el valor de mantener la trazabilidad.
FL-1.4.5	(K2)	Comparar los diferentes roles en la prueba.
1.5. Competencias Esenciales y Buenas Prácticas en la Prueba		
FL-1.5.1	(K2)	Dar ejemplos de las competencias genéricas necesarias para probar.
FL-1.5.2	(K1)	Recordar las ventajas del enfoque de equipo completo.
FL-1.5.3	(K2)	Distinguir las ventajas e inconvenientes de la independencia de la prueba.

1.1 ¿Qué es Probar?

Los sistemas de software son parte integrante de nuestra vida cotidiana. La mayoría de la gente ha tenido experiencias con software que no funcionaba como se esperaba. Un software que no funciona correctamente puede acarrear muchos problemas, como pérdidas económicas, de tiempo o de reputación empresarial y, en casos extremos, incluso lesiones o la muerte. La prueba de software evalúa la calidad del software y ayuda a reducir el riesgo de fallo del software en operación.

La prueba de software es un conjunto de actividades para descubrir defectos y evaluar la calidad de los artefactos de software. Cuando se prueban, estos artefactos se conocen como objetos de prueba. Un concepto erróneo habitual sobre la prueba es que sólo consiste en ejecutar pruebas (es decir, ejecutar el software y comprobar los resultados de las pruebas). Sin embargo, la prueba del software también incluye otras actividades y debe estar alineada con el ciclo de vida de desarrollo del software (véase el capítulo 2).

Otro concepto erróneo habitual sobre la prueba es que ésta se concentra por completo en la verificación del objeto de prueba. Aunque la prueba implica verificación, es decir, comprobar si el sistema cumple los requisitos especificados, también implica validación, lo que significa comprobar si el sistema satisface las necesidades de los usuarios y otros implicados en su entorno operativo.

La prueba puede ser dinámica o estática. La prueba dinámica implica la ejecución del software, mientras que la prueba estática no. La prueba estática incluye revisiones (véase el capítulo 3) y análisis estático. La prueba dinámica utiliza distintos tipos de técnicas de prueba y enfoques de prueba para obtener casos de prueba (véase el capítulo 4).

Probar no es sólo una actividad técnica. También necesita que se planifique, gestione, estime, monitorice y controle de forma adecuada (véase el capítulo 5).

Los probadores utilizan herramientas (véase el capítulo 6), pero es importante recordar que la prueba es en gran medida una actividad intelectual, que requiere que los probadores tengan conocimientos especializados, utilicen competencias analíticas y apliquen el pensamiento crítico y el pensamiento sistémico (Myers 2011, Roman 2018).

El estándar ISO/IEC/IEEE 29119-1 proporciona más información sobre los conceptos de prueba de software.

1.1.1 Objetivos de Prueba

Los objetivos de prueba característicos son:

- Evaluar productos de trabajo como requisitos, historias de usuario, diseños y código.
- Provocar fallos y encontrar defectos.
- Asegurar la cobertura necesaria de un objeto de prueba.
- Reducir el nivel de riesgo asociado a una calidad inadecuada del software.
- Verificar si se han cumplido los requisitos especificados.
- Verificar que un objeto de prueba cumple los requisitos contractuales, legales y normativos.
- Proporcionar información a los implicados para que puedan tomar decisiones informadas.
- Generar confianza en la calidad del objeto de prueba.
- Validar si el objeto de prueba está completo y funciona como esperan los implicados.

Los objetivos de prueba pueden variar en función del contexto, que incluye el producto de trabajo que se está probando, el nivel de prueba, los riesgos, el ciclo de vida de desarrollo del software (SDLC) que se está siguiendo y los factores relacionados con el contexto del negocio, por ejemplo, la estructura corporativa, las consideraciones relativas a la competencia o el tiempo de comercialización.

1.1.2 Probar y Depurar

Probar y depurar son actividades distintas. Probar puede desencadenar fallos causados por defectos en el software (prueba dinámica) o puede, de forma directa, encontrar defectos en el objeto de prueba (prueba estática).

Cuando una prueba dinámica (véase el capítulo 4) desencadena un fallo, la depuración se ocupa de encontrar las causas de este fallo (defectos), analizar estas causas y eliminarlas. En este caso, el proceso típico de depuración implica:

- Reproducción del fallo.
- Diagnóstico (hallazgo de la causa raíz).
- Corrección de la causa.

La prueba de confirmación posterior comprueba si las correcciones han resuelto el problema. Preferiblemente, la prueba de confirmación la realiza la misma persona que llevó a cabo la prueba inicial. También se puede llevar a cabo una prueba de regresión posterior, para comprobar si las correcciones están causando fallos en otras partes del objeto de prueba (véase la sección 2.2.3 para más información sobre la prueba de confirmación y la prueba de regresión).

Cuando la prueba estática identifica un defecto, la depuración se ocupa de eliminarlo. No se necesita reproducción ni diagnóstico, ya que la prueba estática encuentra directamente los defectos y no puede provocar fallos (véase el capítulo 3).

1.2 ¿Por qué es Necesario Probar?

Probar, como forma de control de la calidad, ayuda a alcanzar los objetivos acordados dentro del alcance, el tiempo, la calidad y las limitaciones presupuestarias establecidos. La contribución de la prueba al éxito no debe limitarse a las actividades del equipo de prueba. Cualquier implicado puede utilizar sus competencias en materia de prueba para contribuir al éxito del proyecto. Probar componentes, sistemas y la documentación asociada ayuda a identificar defectos en el software:

1.2.1 Contribuciones de la prueba al éxito

Probar proporciona un medio rentable de detectar defectos. Estos defectos pueden eliminarse posteriormente (mediante la depuración, una actividad ajena a la prueba), por tanto, la prueba contribuye indirectamente a lograr objetos de prueba de mayor calidad.

La prueba proporciona medios para evaluar directamente la calidad de un objeto de prueba en varias etapas del ciclo de vida de desarrollo software CVDS. Estas mediciones se utilizan como parte de una actividad de gestión de proyectos más amplia, contribuyendo a las decisiones para pasar a la siguiente etapa del SDLC, como por ejemplo la decisión de entrega.

Probar proporciona a los usuarios una representación indirecta en el proyecto de desarrollo. Los probadores aseguran que se tiene en cuenta la comprensión de las necesidades de los usuarios a lo largo de todo el ciclo de vida de desarrollo. La alternativa es implicar a un conjunto representativo de usuarios como parte del proyecto de desarrollo, lo que no suele ser posible debido a los elevados costes y a la falta de disponibilidad de usuarios adecuados.

También puede ser necesario realizar pruebas para cumplir requisitos contractuales o legales, o para ajustarse a los estándares reglamentarios.

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

Aunque a menudo se utilizan indistintamente los términos "prueba" y "aseguramiento de la calidad" (AC) , prueba y AC no son lo mismo. La prueba es una forma de control de la calidad (CC.

El CC es un enfoque correctivo orientado al producto que se concentra en aquellas actividades que contribuyen a la consecución de unos niveles de calidad adecuados. La prueba es una de las principales formas de control de la calidad, mientras que otras incluyen métodos formales (comprobación de modelo y prueba de corrección), simulación y creación de prototipos.

El aseguramiento de la calidad es un enfoque preventivo orientado a los procesos que se concentra en la implementación y mejora de los mismos. Funciona sobre la base de que, si un buen proceso se sigue correctamente, entonces generará un buen producto. La garantía de calidad se aplica tanto al proceso de desarrollo como al de prueba, y es responsabilidad de todos los que participan en un proyecto.

Los resultados de la prueba son utilizados por el Aseguramiento de la Calidad (AC) y el Control de la Calidad (CC). En el control de calidad se utilizan para corregir defectos, mientras que en la garantía de calidad proporcionan retroalimentación sobre el rendimiento de los procesos de desarrollo y prueba.

1.2.3 Errores, Defectos, Fallos y Causas Raíz

Los seres humanos cometen errores (equivocaciones), que producen defectos, que a su vez pueden dar lugar a fallos. Los seres humanos cometen errores por diversas razones, como la presión por razones de tiempo, la complejidad de los productos de trabajo, los procesos, la infraestructura o las interacciones, o simplemente porque están cansados o carecen de la formación adecuada.

Los defectos pueden encontrarse en la documentación, como una especificación de requisitos o un guion de prueba, en el código fuente o en un artefacto de apoyo como un archivo de construcción. Los defectos en los artefactos producidos al principio del CVDS, si no se detectan, suelen dar lugar a artefactos defectuosos más adelante en el ciclo de vida. Si se ejecuta un defecto en el código, el sistema puede fallar al hacer lo que debería hacer, o hacer algo que no debería, provocando un fallo. Algunos defectos siempre darán lugar a un fallo si se ejecutan, mientras que otros sólo darán lugar a un fallo en circunstancias específicas, y algunos puede que nunca den lugar a un fallo.

Los errores y los defectos no son la única causa de los fallos. Los fallos también pueden deberse a condiciones ambientales, como cuando la radiación o el campo electromagnético provocan defectos en el firmware.

Una causa raíz es una razón fundamental por la que se produce un problema (por ejemplo, una situación que conduce a un error). Las causas raíz se identifican mediante el análisis de la causa raíz, que suele realizarse cuando se produce un fallo o se identifica un defecto. Se cree que pueden evitarse otros fallos o defectos similares o reducirse su frecuencia tratando la causa raíz, por ejemplo, eliminándola.

1.3 Principios de la Prueba

A lo largo de los años se han sugerido una serie de principios de prueba que ofrecen directrices generales aplicables a todas las pruebas. En este programa de estudio se describen siete de estos principios.

1. **La prueba muestra la presencia de defectos, no su ausencia.** La prueba puede mostrar que hay defectos en el objeto de prueba, pero no puede demostrar que no los haya (Buxton 1970). La prueba reduce la probabilidad de que queden defectos por descubrir en el objeto de prueba, pero aunque no se encuentren defectos, la prueba no puede demostrar la corrección del objeto de prueba.
2. **La prueba exhaustiva es imposible.** Probarlo todo no es factible salvo en casos triviales (Manna 1978). En lugar de intentar probar exhaustivamente, deberían utilizarse técnicas de prueba (véase el capítulo 4), priorización de casos de prueba (véase el apartado 5.1.5) y pruebas basadas en el riesgo (véase el apartado 5.2), para concentrar los esfuerzos de prueba.
3. La prueba temprana ahorra tiempo y dinero. Los defectos que se eliminan al principio del proceso no causarán defectos posteriores en los productos de trabajo derivados. El coste de la calidad se reducirá, ya que se producirán menos fallos más adelante en el CVDS (Boehm 1981). Para encontrar defectos en una fase temprana, tanto las pruebas estáticas (véase el capítulo 3) como las pruebas dinámicas (véase el capítulo 4) deben iniciarse lo antes posible.
4. **Los defectos se agrupan.** Un pequeño número de componentes del sistema suelen contener la mayoría de los defectos descubiertos o son responsables de la mayoría de los fallos operativos (Enders 1975). Este fenómeno es una ilustración del principio de Pareto. Las agrupaciones de defectos previstas, y las agrupaciones de defectos reales observadas durante la prueba o en operación, son una entrada importante para la prueba basada en el riesgo (véase la sección 5.2).
5. **Las pruebas se desgastan.** Si las pruebas se repiten muchas veces, se vuelven cada vez más ineficaces para detectar nuevos defectos (Beizer 1990). Para superar este efecto, puede que sea necesario modificar las pruebas y los datos de prueba existentes, y que haya que escribir nuevas pruebas. Sin embargo, en algunos casos, la repetición de las mismas pruebas puede tener un resultado beneficioso, por ejemplo, en las pruebas de regresión automatizadas (véase la sección 2.2.3).
6. **La prueba depende del contexto.** No existe un único enfoque de prueba aplicable universalmente. La prueba se realiza de forma diferente en distintos contextos (Kaner 2011).
7. **Falacia de la ausencia de defectos.** Es una falacia (es decir, una idea equivocada) esperar que la verificación del software asegure el éxito de un sistema. Probar a fondo todos los requisitos especificados y arreglar todos los defectos encontrados podría seguir produciendo un sistema que no satisface las necesidades y expectativas de los usuarios, que no ayuda a conseguir los objetivos de negocio del cliente y que es inferior en comparación con otros sistemas de la competencia. Además de la verificación, también debe llevarse a cabo la validación (Boehm 1981).

1.4 Actividades de Prueba, Productos de Prueba y Roles de Prueba

La prueba depende del contexto, pero, a un alto nivel, existen conjuntos comunes de actividades de prueba sin los cuales es menos probable que la prueba alcance los objetivos de prueba. Estos conjuntos de actividades de prueba forman un proceso de prueba. El proceso de prueba puede adaptarse a una situación determinada en función de diversos factores. Qué actividades de prueba se incluyen en este proceso de prueba, cómo se implementan y cuándo tienen lugar se decide normalmente como parte de la planificación de prueba para la situación específica (véase la sección 5.1).

Las siguientes secciones describen los aspectos generales de este proceso de prueba en términos de actividades y tareas de prueba, el impacto del contexto, el material de prueba, la trazabilidad entre la base de prueba y el material de prueba, y los roles de prueba.

El estándar ISO/IEC/IEEE 29119-2 proporciona más información sobre los procesos de prueba.

1.4.1 Actividades y Tareas de Prueba

Un proceso de prueba suele constar de los principales grupos de actividades que se describen a continuación. Aunque puede parecer que muchas de estas actividades siguen una secuencia lógica, a menudo se implementan de forma iterativa o en paralelo. Estas actividades de prueba suelen necesitar adaptarse al sistema y al proyecto.

Planificación de la Prueba.

La planificación de la prueba consiste en definir los objetivos de la prueba y, a continuación, seleccionar el enfoque que mejor permita alcanzar los objetivos dentro de las limitaciones impuestas por el contexto general. La planificación de la prueba se explica con más detalle en la sección 5.1.

Monitorización y Control de la Prueba.

La monitorización de la prueba implica la comprobación continua de todas las actividades de la prueba y la comparación del avance real con el plan. El control de prueba implica la adopción de las medidas necesarias para cumplir los objetivos de la prueba. La monitorización de prueba y el control de prueba se explican con más detalle en la sección 5.3.

Análisis de la Prueba.

El análisis de la prueba incluye el análisis de la base de prueba para identificar las prestaciones que pueden ser objeto de prueba y para definir y priorizar las condiciones de prueba asociadas, junto con los riesgos y niveles de riesgo relacionados (véase la sección 5.2). La base de prueba y los objetos de prueba también se evalúan para identificar los defectos que pueden contener y valorar su capacidad de ser probados. El análisis de prueba suele apoyarse en el uso de técnicas de prueba (véase el capítulo 4). El análisis de la prueba responde a la pregunta "¿qué hay que probar?" en términos de criterios de cobertura medibles.

Diseño de la Prueba.

El diseño de prueba incluye la elaboración de las condiciones de prueba en casos de prueba y otros productos de prueba (por ejemplo, contratos de prueba). Esta actividad suele implicar la identificación de elementos de cobertura, que sirven de referencia para especificar las entradas de los casos de prueba. Las técnicas de prueba (véase el capítulo 4) pueden servir de apoyo a esta actividad. El diseño de pruebas también incluye la definición de los requisitos de datos de prueba, el diseño del entorno de prueba y la identificación de cualquier otra infraestructura y herramientas necesarias. El diseño de prueba responde a la pregunta "¿cómo llevar a cabo la prueba?".

Implementación de la Prueba.

La implementación de prueba incluye la creación o la adquisición de los productos de prueba necesarios para la ejecución de prueba (por ejemplo, los datos de prueba). Los casos de prueba pueden organizarse en procedimientos de prueba y suelen reunirse en conjuntos de prueba. Se crean guiones de prueba manuales y automatizados. Se priorizan los procedimientos de prueba y se organizan dentro de un calendario de ejecución de prueba para una ejecución de prueba eficiente (véase la sección 5.1.5). Se construye el entorno de prueba y se verifica que está configurado correctamente.

Ejecución de la Prueba.

La ejecución de la prueba incluye la realización de las pruebas de acuerdo con el calendario de ejecución de prueba (ejecuciones de pruebas). La ejecución de una prueba puede ser manual o automatizada. La ejecución de prueba puede adoptar muchas formas, incluidas la prueba continua o sesiones de prueba en pareja. Los resultados de prueba reales se comparan con los resultados esperados. Se registran los resultados de la prueba. Las anomalías se analizan para identificar sus causas probables. Este análisis permite informar de las anomalías en función de los fallos observados (véase el apartado 5.5).

Compleción de la Prueba.

Las actividades de compleción de la prueba suelen producirse en los hitos del proyecto (por ejemplo, la entrega, el final de una iteración, la compleción del nivel de prueba) para cualquier defecto no resuelto, solicitud de cambio o elemento de lista de trabajo acumulado del producto que se haya creado. Cualquier producto de prueba que pueda ser útil en el futuro se identifica y se archiva o se entrega a los equipos adecuados. Se desactiva el entorno de prueba hasta un estado acordado. Se analizan las actividades de prueba para identificar las lecciones aprendidas y las mejoras para futuras iteraciones, entregas o proyectos (véase la sección 2.1.6). Se crea un informe de compleción de la prueba y se comunica a los implicados.

1.4.2 El Proceso de Prueba en Contexto

La prueba no se realiza de forma aislada. Las actividades de prueba son parte integrante de los procesos de desarrollo que se llevan a cabo en una organización. La prueba también está sufragada por los implicados y su objetivo final es ayudar a satisfacer las necesidades de negocio de los implicados. Por lo tanto, la forma en que se lleve a cabo la prueba dependerá de una serie de factores contextuales entre los que se incluyen:

- Implicados (necesidades, expectativas, requisitos, voluntad de cooperación, etc.)
- Miembros del equipo (competencias, conocimientos, nivel de experiencia, disponibilidad, necesidades de formación, etc.)
- Dominio del negocio (criticidad del objeto de prueba, riesgos identificados, necesidades del mercado, normativa legal específica, etc.)
- Factores técnicos (tipo de software, arquitectura del producto, tecnología utilizada, etc.)
- Restricciones del proyecto (alcance, tiempo, presupuesto, recursos, etc.)
- Factores organizativos (estructura organizativa, políticas existentes, prácticas utilizadas, etc.)
- Ciclo de vida de desarrollo del software (prácticas de ingeniería, métodos de desarrollo, etc.)
- Herramientas (disponibilidad, usabilidad, cumplimiento, etc.)

Estos factores influirán en muchos asuntos relacionados con la prueba, entre ellos: la estrategia de prueba, las técnicas de prueba utilizadas, el grado de automatización de la prueba, el nivel de cobertura requerido, el nivel de detalle de la documentación de prueba, el suministro de información, etc.

1.4.3 Productos de Prueba

El software de prueba se crea como producto de salida de las actividades de prueba descritas en la sección 1.4.1. Existe una variación significativa en la forma en que las distintas organizaciones producen, conforman, nombran, organizan y gestionan sus productos de trabajo. Una gestión de la configuración adecuada (véase la sección 5.4) asegura la consistencia y la integridad de los productos de trabajo. La siguiente lista de productos de trabajo no es exhaustiva:

- **Productos del Trabajo de Planificación de la Prueba**
  - Entre los productos del trabajo de planificación de la prueba se incluyen: el plan de prueba, el calendario de prueba, el registro de riesgos y los criterios de entrada y salida (véase la sección 5.1). El registro de riesgos es una lista de riesgos junto con la probabilidad del riesgo, el impacto del riesgo e información sobre la mitigación del riesgo (véase la sección 5.2). El calendario de prueba, el registro de riesgos y los criterios de entrada y salida suelen formar parte del plan de prueba.
- **Productos de Trabajo de Monitorización y Control de la Prueba**
  - Los productos de trabajo de monitorización y control de la prueba incluyen: informes del avance de la prueba (véase el apartado 5.3.2), documentación de las directivas de control (véase el apartado 5.3) e información sobre los riesgos (véase el apartado 5.2).
- **Productos del Trabajo de Análisis de la Prueba**
  - Los productos del trabajo de análisis de la prueba incluyen: condiciones de prueba (priorizadas) (por ejemplo, criterios de aceptación, véase la sección 4.5.2), e informes de defectos relativos a los defectos en la base de prueba (si no se solucionan directamente).
- **Productos del Trabajo del Diseño de la Prueba**
  - Los productos de trabajo del diseño de prueba incluyen: casos de prueba (priorizados), contratos de prueba, elementos de cobertura, requisitos de datos de prueba y requisitos del entorno de prueba.
- **Productos del Trabajo de Implementación de la Prueba**
  - Los productos de trabajo de implementación de prueba incluyen: procedimientos de prueba, guiones de prueba automatizados, juegos de prueba, datos de prueba, calendario de ejecución de la prueba y elementos del entorno de prueba. Algunos ejemplos de elementos del entorno de pruebas son: stubs, controladores, simuladores y virtualizaciones de servicios.
- **Productos del Trabajo de Ejecución de la Prueba**
  - Los productos de trabajo de la ejecución de la prueba incluyen: registros en bitácora de prueba e informes de defectos (véase la sección 5.5).
- **Productos del Trabajo de Compleción de la Prueba**
  - Los productos de trabajo de la compleción de la prueba incluyen: informe de compleción de la prueba (véase la sección 5.3.2), elementos de acción para la mejora de proyectos o iteraciones posteriores, lecciones aprendidas documentadas y solicitudes de cambio (por ejemplo, como elementos de la lista de trabajo acumulado del producto).

1.4.4 Trazabilidad entre Bases de Prueba y Productos de Prueba

Para implementar una monitorización y un control efectivos de la prueba, es importante establecer y mantener la trazabilidad a lo largo del proceso de prueba entre los elementos de la base de prueba, los productos de prueba asociados a estos elementos (por ejemplo, las condiciones de la prueba, los riesgos, los casos de prueba), los resultados de la prueba y los defectos detectados.

Una trazabilidad precisa soporta la evaluación de la cobertura, por lo que es muy útil que se definan criterios de cobertura medibles en la base de prueba. Los criterios de cobertura pueden funcionar como indicadores clave de rendimiento para impulsar las actividades que muestran hasta qué punto se han alcanzado los objetivos de la prueba (véase la sección 1.1.1). Por ejemplo:

- La trazabilidad de los casos de prueba a los requisitos puede verificar que los requisitos están cubiertos por los casos de prueba.
- La trazabilidad de los resultados de prueba a los riesgos puede utilizarse para evaluar el nivel de riesgo residual en un objeto de prueba.

Además de evaluar la cobertura, una buena trazabilidad permite determinar el impacto de los cambios, facilita las auditorías de la prueba y ayuda a cumplir los criterios de gobernanza de TI. Una buena trazabilidad también facilita la comprensión de los informes de avance y compleción de la prueba al incluir el estado de los elementos de la base de prueba. Esto también puede ayudar a comunicar los aspectos técnicos de la prueba a los implicados de forma comprensible. La trazabilidad proporciona información para evaluar la calidad del producto, la capacidad del proceso y el avance del proyecto con respecto a los objetivos de negocio.

1.4.5 Roles en la Prueba

En este programa de estudio se tratan dos roles principales en la prueba: un rol de gestión de prueba y un rol de prueba. Las actividades y tareas asignadas a estos dos roles dependen de factores como el contexto del proyecto y del producto, las competencias de las personas que desempeñan los roles y la organización.

El rol de gestión de prueba asume la responsabilidad general del proceso de prueba, del equipo de prueba y de la dirección de las actividades de prueba. El rol de gestión de prueba se concentra principalmente en las actividades de planificación de prueba, monitorización y control de prueba y compleción de la prueba. La forma en que se lleva a cabo el rol de gestión de pruebas varía en función del contexto. Por ejemplo, en el desarrollo ágil de software, algunas de las tareas de gestión de pruebas pueden correr a cargo del equipo ágil. Las tareas que abarcan a varios equipos o a toda la organización pueden ser realizadas por jefes de prueba ajenos al equipo de desarrollo.

El rol de prueba asume la responsabilidad general del aspecto asociado a la ingeniería (técnica) de la prueba. El rol de prueba se concentra principalmente en las actividades de análisis de prueba, diseño de prueba, implementación de prueba y ejecución de prueba.

Diferentes personas pueden asumir estos roles en diferentes momentos. Por ejemplo, el rol de gestión de la prueba puede ser desempeñado por un jefe de equipo, por un jefe de prueba, por un jefe de desarrollo, etc. También es posible que una misma persona asuma los roles de prueba y gestión de prueba al mismo tiempo.

1.5 Competencias Esenciales y Buenas Prácticas de la Prueba

La competencia es la capacidad de hacer algo bien que proviene del conocimiento, la práctica y la aptitud de cada uno. Los buenos probadores deben poseer algunas competencias esenciales para hacer bien su trabajo. Los buenos probadores deben ser jugadores de equipo eficaces y deben ser capaces de realizar pruebas en diferentes niveles de independencia de la prueba.

1.5.1 Competencias Genéricas Necesarias para Probar

A pesar de ser genéricas, las siguientes competencias son especialmente relevantes para los probadores:

- Conocimiento en materia de prueba (para aumentar la efectividad de la prueba, por ejemplo, mediante el uso de técnicas de prueba).
- Minuciosidad, cuidado, curiosidad, atención a los detalles, ser metódico (para identificar defectos, especialmente los difíciles de encontrar).
- Buena competencia en el ámbito de la comunicación escucha activa, ser un jugador de equipo (para interactuar eficazmente con todos los implicados, transmitir información a los demás, hacerse entender e informar y discutir los defectos).
- Pensamiento analítico, pensamiento crítico, creatividad (para aumentar la efectividad de las pruebas).
- Conocimientos técnicos (para aumentar la eficiencia de las pruebas, por ejemplo, utilizando herramientas de prueba adecuadas).
- Conocimientos del dominio (para poder comprender y comunicarse con los usuarios finales/representantes de negocio).

A menudo, los probadores son los portadores de las malas noticias. Es un rasgo humano común culpar al portador de malas noticias. Esto hace que las competencias de comunicación sean cruciales para los probadores. Comunicar los resultados de prueba puede percibirse como una crítica al producto y a su autor. El sesgo de confirmación puede dificultar la aceptación de información que discrepa de las creencias actuales. Algunas personas pueden percibir las pruebas como una actividad destructiva, a pesar de que contribuyen en gran medida al éxito del proyecto y a la calidad del producto. Para intentar mejorar esta opinión, la información sobre defectos y fallos debe comunicarse de forma constructiva.

1.5.2 Enfoque de Equipo Completo

Una de las competencias importantes para un probador es la capacidad de trabajar eficazmente en un contexto de equipo y de contribuir positivamente a los objetivos del mismo. El enfoque de equipo completo - una práctica procedente de la Programación Extrema (véase la sección 2.1) - se basa en esta competencia.

En el enfoque de equipo completo, cualquier miembro del equipo con los conocimientos y competencias necesarios puede realizar cualquier tarea, y todos son responsables de la calidad. Los miembros del equipo comparten el mismo espacio de trabajo (físico o virtual), ya que la ubicación común facilita la comunicación y la interacción. El enfoque de equipo completo mejora la dinámica de equipo, potencia la comunicación y la colaboración dentro del equipo y crea sinergia al permitir que los diversos conjuntos de competencias dentro del equipo se aprovechen en beneficio del proyecto.

Los probadores trabajan en estrecha colaboración con otros miembros del equipo para asegurar que se alcanzan los niveles de calidad deseados. Esto incluye colaborar con los representantes de negocio para ayudarles a crear pruebas de aceptación adecuadas y trabajar con los desarrolladores para acordar la estrategia de prueba y decidir los enfoques de automatización de la prueba. De este modo, los probadores pueden transferir conocimientos sobre la prueba a otros miembros del equipo e influir en el desarrollo del producto.

Dependiendo del contexto, el enfoque de equipo completo puede no ser siempre apropiado. Por ejemplo, en algunas situaciones, como las críticas para la seguridad, puede necesitarse un alto nivel de independencia en la prueba.

1.5.3 Independencia de la Prueba

Un cierto grado de independencia hace que el probador sea más eficaz a la hora de encontrar defectos debido a las diferencias entre los sesgos cognitivos del autor y del probador (cf. Salman 1995). Sin embargo, la independencia no sustituye a la familiaridad; por ejemplo, los desarrolladores pueden encontrar eficazmente muchos defectos en su propio código.

Los productos del trabajo pueden ser probados por su autor (sin independencia), por los compañeros del autor del mismo equipo (cierta independencia), por probadores ajenos al equipo del autor pero dentro de la organización (alta independencia), o por probadores ajenos a la organización (independencia muy alta). Para la mayoría de los proyectos, suele ser mejor llevar a cabo las pruebas con varios niveles de independencia (por ejemplo, que los desarrolladores realicen las pruebas de integración de componentes, que el equipo de pruebas realice las pruebas de integración de sistemas y sistemas, y que los representantes de negocio realicen las pruebas de aceptación).

La principal ventaja de la independencia de la prueba es que es probable que los probadores independientes reconozcan diferentes tipos de fallos y defectos en comparación con los desarrolladores debido a sus diferentes antecedentes, perspectivas técnicas y sesgos. Además, un probador independiente puede verificar, cuestionar o refutar las suposiciones realizadas por los implicados durante la especificación e implementación del sistema.

Sin embargo, también existen algunos inconvenientes. Los probadores independientes pueden estar aislados del equipo de desarrollo, lo que puede provocar una falta de colaboración, problemas de comunicación o una relación de confrontación con el equipo de desarrollo. Los desarrolladores pueden perder el sentido de la responsabilidad por la calidad. Los probadores independientes pueden ser vistos como un cuello de botella o culpables de los retrasos en la entrega.

2 Prueba a lo Largo del Ciclo de Vida de Desarrollo de Software - 130 minutos

Duración: 130 minutos

Palabras Clave[2]

Español	Inglés
prueba de aceptación	acceptance testing
prueba de caja negra	black-box testing
prueba de integración de componentes	component integration testing
prueba de componentes	component testing
prueba de confirmación	confirmation testing
prueba funcional	functional testing
prueba de integración	integration testing
prueba de mantenimiento	maintenance testing
prueba no funcional	non-functional testing
prueba no funcional	regression testing
desplazamiento a la izquierda	shift-left
prueba de integración de sistemas	system integration testing
prueba de sistema	system testing
nivel de prueba	test level
objeto de prueba	test object
tipo de prueba	test type
prueba de caja blanca	white-box testing

Objetivos de Aprendizaje para "Capítulo 2"

2.1 La prueba en el Contexto de un Ciclo de Vida de Desarrollo de Software

FL-2.1.1	(K2)	Explicar el impacto del ciclo de vida de desarrollo de software elegido en la prueba.
FL-2.1.2	(K1)	Recordar las buenas prácticas de prueba que se aplican a todos los ciclos de vida de desarrollo del software.
FL-2.1.3	(K1)	Recordar los ejemplos de enfoques de prueba primero para el desarrollo.
FL-2.1.4	(K2)	Resumir cómo DevOps puede tener un impacto en la prueba.
FL-2.1.5	(K2)	Explicar el enfoque de desplazamiento a la izquierda.

[2] Las palabras clave se encuentran ordenadas por orden alfabético de los términos en inglés.

FL-2.1.6	(K2)	Explicar cómo se pueden utilizar las retrospectivas como mecanismo para la mejora del proceso.
2.2 Niveles de Prueba y Tipos de Prueba		
FL-2.2.1	(K2)	Distinguir los diferentes niveles de prueba.
FL-2.2.2	(K2)	Distinguir los diferentes tipos de prueba.
FL-2.2.3	(K2)	Distinguir la prueba de confirmación de la prueba de regresión.
2.3 Prueba de Mantenimiento		
FL-2.3.1	(K2)	Resumir la prueba de mantenimiento y sus desencadenantes.

2.1 La Prueba en el Contexto de un Ciclo de Vida de Desarrollo de Software

Un modelo de ciclo de vida de desarrollo de software (CVDS) es una representación abstracta y de alto nivel del proceso de desarrollo de software. Un modelo CVDS define cómo se relacionan entre sí, tanto lógica como cronológicamente, las diferentes fases de desarrollo y los tipos de actividades que se realizan dentro de este proceso. Algunos ejemplos de modelos de CVDS son: los modelos de desarrollo secuencial (por ejemplo, el modelo en cascada, el modelo V), los modelos de desarrollo iterativo (por ejemplo, el modelo en espiral, el prototipado) y los modelos de desarrollo incremental (por ejemplo, el Proceso Unificado).

Algunas actividades de los procesos de desarrollo de software también pueden describirse mediante métodos de desarrollo de software más detallados y prácticas Ágiles. Algunos ejemplos son: el desarrollo guiado por pruebas de aceptación (DGPA), el desarrollo guiado por el comportamiento (DGC), el diseño guiado por dominios (DGD), la programación extrema (XP), el desarrollo guiado por prestaciones (DGP), Kanban, Lean IT, Scrum y el desarrollo guiado por pruebas (DGP).

2.1.1 Impacto del Ciclo de Vida de Desarrollo de Software en la Prueba

La prueba debe adaptarse al CVDS para tener éxito. La elección del CVDS repercute en:

- Alcance y cronología de las actividades de prueba (por ejemplo, niveles de prueba y tipos de prueba).
- Nivel de detalle de la documentación de prueba.
- Elección de técnicas de prueba y enfoque de prueba.
- Alcance de la automatización de la prueba.
- Rol y responsabilidades de un probador.

En las fases iniciales de los modelos de desarrollo secuencial, los probadores suelen participar en la revisión de requisitos, el análisis de prueba y el diseño de prueba. El código ejecutable suele crearse en las fases posteriores, por lo que normalmente la prueba dinámica no puede realizarse al principio del CVDS.

En algunos modelos de desarrollo iterativo e incremental, se supone que cada iteración entrega un prototipo funcional o un incremento del producto. Esto implica que en cada iteración se puede probar tanto estática como dinámicamente en todos los niveles de prueba. La entrega frecuente de incrementos requiere una retroalimentación rápida y pruebas de regresión extensivas.

El desarrollo de software Ágil asume que pueden producirse cambios a lo largo de todo el proyecto. Por lo tanto, en los proyectos ágiles se favorece una documentación de producto de trabajo ligera y una amplia automatización de la prueba para facilitar la prueba de regresión. Además, la mayor parte de las pruebas manuales suelen realizarse mediante técnicas de prueba basadas en la experiencia (véase la sección 4.4) que no requieren un análisis y diseño de prueba previos exhaustivos.

2.1.2 Ciclo de Vida de Desarrollo de Software y Buenas Prácticas de Prueba

Las buenas prácticas de prueba, independientemente del modelo de CVDS elegido, incluyen los siguientes elementos:

- Cada actividad de desarrollo de software tiene su correspondiente actividad de prueba, de modo que todas las actividades de desarrollo están sujetas a un control de la calidad.

- Cada nivel de prueba (véase el capítulo 2.2.1) tiene objetivos de prueba específicos y diferentes, lo que permite que las pruebas sean lo suficientemente comprensivas a la vez que se evita la redundancia.
- El análisis y diseño de prueba para un nivel de prueba determinado comienza durante la fase de desarrollo correspondiente del CVDS, para que la prueba pueda atenerse al principio de prueba temprana (véase la sección 1.3).
- Los probadores participan en la revisión de los productos de trabajo tan pronto como están disponibles los borradores de esta documentación, para que esta prueba temprana y la detección de defectos puedan apoyar la estrategia de desplazamiento a la izquierda (véase la sección 2.1.5).

2.1.3 La Prueba como Impulsor del Desarrollo de Software

Desarrollo guiado por prueba (DGP), desarrollo guiado por prueba de aceptación (DGPA) y desarrollo guiado por el comportamiento (DGC son enfoques de desarrollo similares, en los que las pruebas se definen como un medio para conducir el desarrollo. Cada uno de estos enfoques aplica el principio de la prueba temprana (véase el apartado 1.3) y sigue un planteamiento de desplazamiento a la izquierda (véase el apartado 2.1.5), ya que las pruebas se definen antes de escribir el código. Apoyan un modelo de desarrollo iterativo. Estos enfoques se caracterizan por lo siguiente:

Desarrollo Guiado por Prueba (DGP):

- Dirige la codificación mediante casos de prueba (en lugar de un diseño extensivo del software) (Beck 2003).
- Primero se redactan las pruebas, luego se elabora el código para que satisfaga las pruebas y, por último, se refactorizan las pruebas y el código.

Desarrollo Guiado por Prueba de Aceptación (DGPA) (véase el apartado 4.5.3):

- Obtiene pruebas a partir de criterios de aceptación como parte del proceso de diseño del sistema (Gärtner 2011).
- Las pruebas se redactan antes de que la parte de la aplicación se desarrolle para satisfacer las pruebas.

Desarrollo Guiado por Comportamiento (DGC):

- Expresa el comportamiento deseado de una aplicación con casos de prueba escritos en una forma sencilla de lenguaje natural, que es fácil de entender por los implicados - normalmente utilizando el formato Dado/Cuando/Entonces (“Given/When/Then”). (Chelimsky 2010)
- A continuación, los casos de prueba se traducen automáticamente en pruebas ejecutables.

En todos los enfoques anteriores, las pruebas pueden persistir como pruebas automatizadas para asegurar la calidad del código en futuras adaptaciones / refactorizaciones.

2.1.4 DevOps y la Prueba

DevOps es un enfoque organizativo que pretende crear sinergias haciendo que el desarrollo (incluidas las pruebas) y las operaciones trabajen juntos para alcanzar un conjunto de objetivos comunes. DevOps requiere un cambio cultural dentro de una organización para salvar las distancias entre el desarrollo (incluidas las pruebas) y las operaciones, tratando sus funciones con el mismo valor. DevOps promueve la autonomía de los equipos, la retroalimentación rápida, las cadenas de herramientas integradas y prácticas técnicas como la integración continua (IC) y la entrega continua (EC). Esto permite a los equipos

construir, probar y entregar código de alta calidad más rápidamente a través de un canal de entrega DevOps (Kim 2016).

Desde el punto de vista de la prueba, algunas de las ventajas de DevOps son:

- Rápida retroalimentación sobre la calidad del código y sobre si los cambios afectan negativamente al código existente.
- La integración continua IC promueve un enfoque de desplazamiento a la izquierda en la prueba (véase la sección 2.1.5) animando a los desarrolladores a suministrar código de alta calidad acompañado de pruebas de componentes y análisis estático.
- Promueve procesos automatizados como CI/CD que facilitan el establecimiento de entornos de prueba estables.
- Aumenta la visión sobre las características de calidad no funcionales (por ejemplo, rendimiento, fiabilidad).
- La automatización a través de un canal de entrega reduce la necesidad de pruebas manuales repetitivas.
- El riesgo en la regresión se minimiza gracias a la escala y el alcance de las pruebas de regresión automatizadas.

DevOps no está exento de riesgos y desafíos, entre los que se incluyen:

- La tubería[3] de entrega DevOps debe ser definida y establecida.
- Las herramientas CI / CD deben ser introducidas y mantenidas.
- La automatización de la prueba requiere recursos adicionales y puede ser difícil de establecer y mantener.

Aunque DevOps llega con un alto nivel de pruebas automatizadas, la prueba manual - especialmente desde la perspectiva del usuario - seguirá siendo necesaria.

2.1.5 Enfoque “Desplazamiento a la Izquierda”

El principio de prueba temprana (véase la sección 1.3) se denomina a veces desplazamiento a la izquierda porque es un enfoque en el que la prueba se realiza de forma temprana en el CVDS. El desplazamiento a la izquierda normalmente sugiere que la prueba debería realizarse antes (por ejemplo, no esperar a que se implemente el código o a que se integren los componentes), pero no significa que deba descuidarse la prueba más avanzada en el CVDS.

Existen algunas buenas prácticas que ilustran cómo lograr un "desplazamiento a la izquierda" en la prueba, entre las que se incluyen:

- Revisión de la especificación desde la perspectiva de la prueba. Estas actividades de revisión de las especificaciones suelen encontrar defectos potenciales, como ambigüedades, incompletitud e inconsistencias.
- Redactar casos de prueba desde antes de escribir el código y hacer que éste se ejecute en un arnés de prueba durante la implementación de código.

[3] Buscar/Confirmar traducción del término

- Utilizar integración continua (IC) e incluso mejor entrega continua (EC), ya que aporta una retroalimentación rápida y pruebas de componentes automatizadas para acompañar al código fuente cuando se envía al repositorio de código.
- Completar el análisis estático del código fuente antes de la prueba dinámica, o como parte de un proceso automatizado.
- Realizar pruebas no funcionales empezando en el nivel de prueba de componente, siempre que sea posible. Se trata de una forma de desplazamiento a la izquierda, ya que estos tipos de pruebas no funcionales tienden a realizarse más adelante en el CVDS, cuando se dispone de un sistema completo y de un entorno de prueba representativo.

Un enfoque de desplazamiento a la izquierda puede dar lugar a formación, esfuerzo y/o costes adicionales al principio del proceso, pero se espera que ahorre esfuerzos y/o costes con posterioridad.

Para el enfoque de desplazamiento a la izquierda es importante que los implicados estén convencidos y asuman este concepto.

2.1.6 Retrospectivas y Mejora de Proceso

Las retrospectivas (también conocidas como "reuniones post proyecto" y retrospectivas de proyecto) suelen celebrarse al final de un proyecto o de una iteración, en un hito de entrega, o pueden celebrarse cuando se necesiten. La cronología y la organización de las retrospectivas dependen del modelo de CVDS concreto que se siga. En estas reuniones los participantes (no sólo probadores, sino también, por ejemplo, desarrolladores, arquitectos, propietario de producto, analistas de negocio) debaten:

- ¿Qué tuvo éxito y debe conservarse?
- ¿Qué no tuvo éxito y puede mejorarse?
- ¿Cómo incorporar las mejoras y conservar los éxitos en el futuro?

Los resultados deben registrarse y normalmente forman parte del informe de compleción de la prueba (véase la sección 5.3.2). Las retrospectivas son fundamentales para el éxito de la implementación de la mejora continua y es importante que se haga un seguimiento de cualquier mejora recomendada.

Entre los beneficios habituales para la prueba se incluyen:

- Aumento de la efectividad / eficiencia de la prueba (por ejemplo, mediante la implementación de sugerencias para la mejora del proceso).
- Aumento de la calidad de los productos de prueba (por ejemplo, mediante la revisión conjunta de los procesos de prueba)
- Unión y aprendizaje en equipo (por ejemplo, como resultado de la oportunidad de plantear problemas y proponer puntos de mejora)
- Mejora de la calidad de la base de prueba (por ejemplo, al poder abordar y solucionar las deficiencias en la extensión y la calidad de los requisitos)
- Mejor cooperación entre el desarrollo y la prueba (por ejemplo, ya que la colaboración se revisa y optimiza con regularidad).

2.2 Niveles de Prueba y Tipos 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, realizado en relación con el software en una etapa determinada de desarrollo, desde componentes individuales hasta sistemas completos o, en su caso, sistemas de sistemas.

Los niveles de prueba están relacionados con otras actividades dentro del CVDS. En los modelos secuenciales de CVDS, los niveles de prueba suelen definirse de tal forma que los criterios de salida de un nivel forman parte de los criterios de entrada del siguiente. En algunos modelos iterativos, esto puede no aplicarse. Las actividades de desarrollo pueden abarcar varios niveles de prueba. Los niveles de prueba pueden solaparse en el tiempo.

Los tipos de prueba son grupos de actividades de prueba relacionadas con características de calidad específicas y la mayoría de esas actividades de prueba pueden realizarse en todos los niveles de prueba.

2.2.1 Niveles de Prueba

En este programa de estudio se describen los siguientes cinco niveles de prueba:

Prueba de Componente

La prueba de componente (también conocida como prueba unitaria) se concentra en probar componentes de forma aislada. A menudo requiere un soporte específico, como arneses de prueba o marcos de trabajo para la prueba unitaria. En general, los desarrolladores realizan la prueba de componente en sus entornos de desarrollo.

Prueba de Integración de Componentes

La prueba de integración de componentes (también conocida como prueba de integración de unidades) se concentra en probar las interfaces y las interacciones entre los componentes. La prueba de integración de componentes depende en gran medida de los enfoques de la estrategia de integración, como ascendente, descendente o big-bang.

Prueba de Sistema

La prueba de sistema se concentra en el comportamiento general y las capacidades de todo un sistema o producto, incluyendo a menudo la prueba funcional de las tareas de extremo a extremo y la prueba no funcional de las características de calidad. Para algunas características de calidad no funcionales, es preferible probarlas en un sistema completo en un entorno de prueba representativo (por ejemplo, la usabilidad). También es posible utilizar simulaciones de subsistemas. Un equipo de prueba independiente puede realizar la prueba de sistema, que está relacionada con las especificaciones del sistema.

Prueba de Integración de Sistemas

La prueba de integración de sistemas se concentra en probar las interfaces del sistema sujeto a prueba y otros sistemas y servicios externos. La prueba de integración de sistemas requiere entornos de prueba adecuados, preferiblemente similares al entorno de operaciones.

Prueba de Aceptación

La prueba de aceptación se concentra en la validación y en demostrar la preparación para el despliegue, lo que significa que el sistema satisface las necesidades de negocio del usuario. Lo ideal es que la prueba de aceptación la realicen usuarios a los que está dirigido. Las principales formas de pruebas de aceptación son: prueba de aceptación de usuario (PAU), prueba de aceptación operativa, prueba de aceptación contractual y de regulación, prueba alfa y prueba beta.

Con el fin de evitar el solapamiento de las actividades de prueba, los niveles de prueba se distinguen por la siguiente lista no exhaustiva de atributos:

- Objeto de prueba.
- Objetivos de prueba.
- Base de prueba.
- Defectos y fallos.
- Enfoque y responsabilidades.

2.2.2 Tipos de Prueba

Existen muchos tipos de prueba que pueden aplicarse en los proyectos. En este programa de estudio se abordan los siguientes cuatro tipos de prueba:

Prueba Funcional

La prueba funcional evalúa las funciones que debe realizar un componente o sistema. Las funciones son " aquello" que debe hacer el objeto de prueba. El objetivo principal de la prueba funcional es comprobar la completitud funcional, la corrección funcional y la pertinencia funcional.

Prueba No Funcional

La prueba no funcional evalúa atributos distintos de las características funcionales de un componente o sistema. La prueba no funcional es la comprobación de "lo bien que se comporta el sistema". El principal objetivo de la prueba no funcional es comprobar las características de calidad del software no funcionales. El estándar ISO/IEC 25010 proporciona la siguiente clasificación de las características de calidad del software no funcionales:

- Eficiencia de desempeño.
- Compatibilidad.
- Usabilidad.
- Fiabilidad.
- Seguridad.
- Mantenibilidad.
- Portabilidad.

A veces es conveniente que la prueba no funcional comience al principio del ciclo de vida (por ejemplo, como parte de las revisiones y prueba de componente o prueba de sistema). Muchas pruebas no funcionales se obtienen a partir de pruebas funcionales, ya que utilizan las mismas pruebas funcionales, pero comprueban que, al realizar la función, se satisface una restricción no funcional (por ejemplo, comprobar que una función se lleva a cabo en un tiempo determinado o que una función puede portarse a una nueva plataforma). El descubrimiento tardío de defectos no funcionales puede suponer una grave amenaza para el éxito de un proyecto. A veces, la prueba no funcional necesita un entorno de prueba muy específico, como un laboratorio de usabilidad para probar la usabilidad.

Prueba de Caja Negra

La prueba de caja negra (véase el apartado 4.2) se basa en la especificación y obtiene pruebas a partir de documentación externa al objeto de prueba. El objetivo principal de la prueba de caja negra es comprobar el comportamiento del sistema frente a sus especificaciones.

Prueba de Caja Blanca

La prueba de caja blanca (véase el apartado 4.3) se basa en la estructura y obtiene pruebas de la implementación o la estructura interna del sistema (por ejemplo, el código, la arquitectura, los flujos de trabajo y los flujos de datos). El objetivo principal de la prueba de caja blanca es que las pruebas cubran la estructura subyacente hasta un nivel aceptable.

Los cuatro tipos de prueba mencionados pueden aplicarse a todos los niveles de prueba, aunque el enfoque será diferente en cada nivel. Se pueden utilizar diferentes técnicas de prueba para obtener condiciones y casos de prueba para todos los tipos de prueba mencionados.

2.2.3 Prueba de Confirmación y Prueba de Regresión

Normalmente se realizan cambios en un componente o sistema para mejorarlo añadiendo una nueva prestación o para solucionarlo eliminando un defecto. La prueba debería entonces incluir también la prueba de confirmación y la prueba de regresión.

La prueba de confirmación confirma que un defecto original se ha corregido con éxito. En función del riesgo, se puede probar la versión corregida del software de varias maneras, entre ellas:

- ejecutando todos los casos de prueba que hayan fallado previamente debido al defecto, o, también mediante
- añadiendo nuevas pruebas para cubrir cualquier cambio que se haya necesitado para arreglar el defecto.

Sin embargo, cuando se dispone de poco tiempo o dinero para solucionar los defectos, la prueba de confirmación puede limitarse a practicar simplemente los pasos que deberían reproducir el fallo causado por el defecto y comprobar que éste no se produce.

La prueba de regresión confirma que no se han producido consecuencias adversas a causa de un cambio, incluida una corrección que ya ha sido sometida a una prueba de confirmación. Estas consecuencias adversas podrían afectar al mismo componente en el que se realizó el cambio, a otros componentes del mismo sistema o incluso a otros sistemas conectados. La prueba de regresión puede no limitarse al objeto de prueba en sí, sino que también puede estar relacionada con el entorno. Es aconsejable realizar primero un análisis de impacto para optimizar el alcance de la prueba de regresión. El análisis de impacto muestra qué partes del software podrían verse afectadas.

Los juegos de pruebas de regresión se ejecutan muchas veces y, por lo general, el número de casos de prueba de regresión aumentará con cada iteración o entrega, por lo que la prueba de regresión es una firme candidata a la automatización. La automatización de estas pruebas debe comenzar en las primeras fases del proyecto. Cuando se utiliza IC, como en DevOps (véase la sección 2.1.4), es una buena práctica incluir también pruebas de regresión automatizadas. Dependiendo de la situación, esto puede incluir pruebas de regresión en diferentes niveles.

Se necesita una prueba de confirmación y/o una prueba de regresión para el objeto de prueba en todos los niveles de prueba si se corrigen defectos y/o se realizan cambios en estos niveles de prueba.

2.3 Prueba de Mantenimiento

Existen diferentes categorías de mantenimiento, puede ser correctivo, adaptable a los cambios del entorno o mejorar el rendimiento o la mantenibilidad (véase ISO/IEC 14764 para más detalles), por lo que el mantenimiento puede implicar entregas/despliegues planificados y entregas/despliegues no planificados (correcciones en caliente). El análisis de impacto puede realizarse antes de realizar un cambio, para ayudar a decidir si el cambio debe realizarse, basándose en las consecuencias potenciales en otras áreas del sistema. Probar los cambios de un sistema en producción incluye tanto la evaluación del éxito de la implementación del cambio como la comprobación de posibles regresiones en las partes del sistema que permanecen inalteradas (que suele ser la mayor parte del sistema).

El alcance de las pruebas de mantenimiento suele depender de:

- El grado de riesgo del cambio.
- El tamaño del sistema existente.
- La magnitud del cambio.

Los desencadenantes del mantenimiento y de la prueba de mantenimiento pueden clasificarse como sigue:

- Modificaciones, como mejoras planificadas (es decir, basadas en la entrega), cambios correctivos o correcciones en caliente.
- Actualizaciones o migraciones del entorno de operación, como de una plataforma a otra, que pueden requerir pruebas asociadas con el nuevo entorno así como del software modificado, o pruebas de conversión de datos cuando se migran datos de otra aplicación al sistema que se está manteniendo.
- Retirada, tal como ocurre cuando una aplicación llega al final de su vida útil. Cuando se retira un sistema, puede ser necesario probar el archivo de datos si se requieren largos periodos de retención de datos. También puede ser necesario probar los procedimientos de recuperación y restauración tras el archivado en caso de que se necesiten determinados datos durante el periodo de archivado.

3 Prueba Estática - 80 minutos

Duración: 80 minutos

Palabras Clave[4]

Español	Inglés
anomalía	anomaly
prueba dinámica	dynamic testing
revisión formal	formal review
revisión informal	informal review
inspección	inspection
revisión	review
análisis estático	static analysis
prueba estática	static testing
revisión técnica	technical review
revisión guiada	walkthrough

Objetivos de Aprendizaje para "Capítulo 3"

3.1 Prueba estática - Fundamentos		
FL-3.1.1	(K1)	Reconocer los tipos de productos que pueden ser evaluados mediante las diferentes técnicas de prueba estática.
FL-3.1.2	(K2)	Explicar el valor de la prueba estática.
FL-3.1.3	(K2)	Comparar y contrastar la prueba estática y la prueba dinámica.
3.2 Retroalimentación y Proceso de Revisión		
FL-3.2.1	(K1)	Identificar las ventajas de una retroalimentación temprana y frecuente de los implicados.
FL-3.2.2	(K2)	Resumir las actividades del proceso de revisión.
FL-3.2.3	(K1)	Recordar qué responsabilidades se asignan a los principales roles a la hora de realizar revisiones.
FL-3.2.4	(K2)	Comparar y contrastar los diferentes tipos de revisión.
FL-3.2.5	(K1)	Recordar los factores que contribuyen al éxito de una revisión.

[4] Las palabras clave se encuentran ordenadas por orden alfabético de los términos en inglés.

3.1 Prueba Estática - Fundamentos

A diferencia de la prueba dinámica, en la prueba estática no es necesario ejecutar el software sujeto a prueba. El código, la especificación del proceso, la especificación de la arquitectura del sistema u otros productos de trabajo se evalúan mediante un examen manual (por ejemplo, revisiones) o con la ayuda de una herramienta (por ejemplo, el análisis estático). Los objetivos de prueba incluyen la mejora de la calidad, la detección de defectos y la evaluación de características como la legibilidad, la completitud, la corrección, la capacidad de ser probado y la consistencia. La prueba estática puede aplicarse tanto para la verificación como para la validación.

Los probadores, los representantes de negocio y los desarrolladores trabajan juntos durante los mapeos de ejemplos, la redacción colaborativa de historias de usuario y las sesiones de refinamiento de la lista de tareas pendientes para asegurar que las historias de usuario y los productos de trabajo relacionados cumplen los criterios definidos, por ejemplo, la definición de preparado (véase la sección 5.1.3). Pueden aplicarse técnicas de revisión para asegurar que las historias de usuario están completas y son comprensibles y que incluyen criterios de aceptación comprobables. Haciendo las preguntas adecuadas, los probadores exploran, cuestionan y ayudan a mejorar las historias de usuario propuestas.

El análisis estático puede identificar problemas antes de la prueba dinámica y suele requerir un menor esfuerzo, ya que no se necesitan casos de prueba y suelen utilizarse herramientas (véase el capítulo 6). El análisis estático suele incorporarse a los marcos de IC (véase la sección 2.1.4). Aunque se utiliza en gran medida para detectar defectos específicos del código, el análisis estático también se emplea para evaluar la mantenibilidad y la seguridad. Los comprobadores ortográficos y las herramientas de legibilidad son otros ejemplos de herramientas de análisis estático.

3.1.1 Productos de Trabajo Susceptibles de Ser Examinados Mediante Prueba Estática

Casi cualquier producto de trabajo puede examinarse mediante una prueba estática. Algunos ejemplos son los documentos de especificación de requisitos, el código fuente, los planes de prueba, los casos de prueba, los elementos de la lista de trabajo acumulado del producto, los contratos de prueba, la documentación del proyecto y los modelos.

Cualquier producto de trabajo que pueda leerse y comprenderse puede ser objeto de una revisión. Sin embargo, para el análisis estático, los productos de trabajo necesitan una estructura con la que puedan ser comprobados (por ejemplo, modelos, código o texto con una sintaxis formal).

Los productos de trabajo que no son apropiados para las pruebas estáticas incluyen aquellos que son difíciles de interpretar por los seres humanos y que no deben ser analizados por herramientas (por ejemplo, el código ejecutable de terceros debido a razones legales).

3.1.2 Valor de la Prueba Estática

La prueba estática puede detectar defectos en las fases más tempranas del CVDS, cumpliendo el principio de la prueba temprana (véase la sección 1.3). También puede identificar defectos que no pueden detectarse mediante pruebas dinámicas (por ejemplo, código inaccesible, patrones de diseño no implementados como se deseaba, defectos en productos de trabajo no ejecutables).

La prueba estática permite evaluar la calidad de los productos de trabajo y generar confianza en ellos. Al verificar los requisitos documentados, los implicados también pueden asegurarse de que dichos requisitos describen sus necesidades reales. Dado que la prueba estática puede realizarse en una fase temprana del CVDS, puede crearse un entendimiento compartido entre los implicados. También mejorará la

comunicación entre los implicados. Por este motivo, se recomienda involucrar a una amplia variedad de implicados en la prueba estática.

Aunque la implementación de las revisiones puede resultar costosa, los costes totales del proyecto suelen ser mucho menores que cuando no se realizan revisiones, ya que se necesita dedicar menos tiempo y esfuerzo a la corrección de defectos cuando el proyecto se encuentra más avanzado.

Los defectos de código pueden detectarse utilizando el análisis estático de forma más eficiente que en la prueba dinámica, lo que suele dar lugar tanto a un menor número de defectos de código como a un menor esfuerzo general de desarrollo.

3.1.3 Diferencias entre la Prueba Estática y la Prueba Dinámica

La prueba estática y la prueba dinámica son prácticas complementarias. Tienen objetivos similares, como apoyar la detección de defectos en los productos de trabajo (véase la sección 1.1.1), pero también existen algunas diferencias, como:

- Tanto la prueba estática como la dinámica (con análisis de fallos) pueden conducir a la detección de defectos; sin embargo, hay algunos tipos de defectos que sólo pueden encontrarse mediante pruebas estáticas o dinámicas.
- La prueba estática encuentra los defectos directamente, mientras que la prueba dinámica provoca fallos a partir de los cuales se determinan los defectos asociados mediante un análisis posterior.
- La prueba estática puede detectar más fácilmente los defectos que se encuentran en caminos a través del código que rara vez se ejecutan o que son difíciles de alcanzar mediante la prueba dinámica.
- La prueba estática puede aplicarse a productos de trabajo no ejecutables, mientras que la prueba dinámica sólo puede aplicarse a productos de trabajo ejecutables.
- La prueba estática se puede utilizar para medir las características de calidad que no dependen de la ejecución del código (por ejemplo, la mantenibilidad), mientras que la prueba dinámica se puede utilizar para medir las características de calidad que dependen de la ejecución del código (por ejemplo, la eficiencia de rendimiento).

Entre los defectos típicos que son más fáciles y/o más económicos de encontrar mediante la prueba estática se incluyen:

Defectos en los requisitos (por ejemplo, inconsistencias, ambigüedades, contradicciones, omisiones, imprecisiones, duplicaciones).

- Defectos de diseño (por ejemplo, estructuras de bases de datos ineficaces, modularidad deficiente).
- Ciertos tipos de defectos de codificación (por ejemplo, variables con valores indefinidos, variables no declaradas, código inalcanzable o duplicado, excesiva complejidad del código).
- Desviaciones de los estándares (por ejemplo, falta de adherencia a las convenciones de nomenclatura en los estándares de codificación).
- Especificaciones incorrectas de la interfaz (por ejemplo, número, tipo u orden de los parámetros no coincidentes).
- Tipos específicos de vulnerabilidades de seguridad (por ejemplo, desbordamientos de memoria intermedia).

- Lagunas o imprecisiones en la cobertura de la base de prueba (por ejemplo, falta de pruebas para un criterio de aceptación).

3.2 Retroalimentación y Proceso de Revisión

3.2.1 Beneficios de una Retroalimentación Temprana y Frecuente de los Implicados

La retroalimentación temprana y frecuente permite la comunicación temprana de posibles problemas de calidad. Si hay poca implicación por parte de los implicados durante el CVDS, es posible que el producto que se desarrolle no cumpla la visión original o actual del implicado. Un fallo en la entrega de lo que la parte interesada desea puede dar lugar a una costosa repetición del trabajo, incumplimiento de los plazos, juegos de culpas e incluso podría llevar al fracaso total del proyecto.

La retroalimentación frecuente por parte de los implicados a lo largo del SDLC puede evitar malentendidos sobre los requisitos y asegurar que los cambios en los requisitos se entiendan y se implementen antes. Esto ayuda al equipo de desarrollo a mejorar su comprensión de lo que están construyendo. Les permite concentrarse en aquellas prestaciones que aportan más valor a los implicados y que tienen un impacto más positivo en los riesgos identificados.

3.2.2 Actividades del Proceso de Revisión

El estándar ISO/IEC 20246 define un proceso de revisión genérico que proporciona un marco estructurado pero flexible a partir del cual se puede adaptar un proceso de revisión específico a una situación concreta. Si la revisión requerida es más formal, entonces se necesitarán más de las tareas descritas para las distintas actividades.

El tamaño de muchos productos de trabajo hace que sean demasiado grandes para ser cubiertos por una sola revisión. El proceso de revisión puede invocarse un par de veces para completar la revisión de todo el producto de trabajo.

Las actividades del proceso de revisión son:

Planificación.

Durante la fase de planificación se definirá el alcance de la revisión, que comprende el propósito, el producto de trabajo que se revisará, las características de calidad que se evaluarán, las áreas en las que se concentrará, los criterios de salida, la información de apoyo como los estándares, el esfuerzo y los plazos de la revisión.

Inicio de la Revisión.

Durante el inicio de la revisión, el objetivo es asegurarse de que todos y cada uno de los implicados están preparados para comenzar la revisión. Esto incluye asegurarse de que cada participante tiene acceso al producto del trabajo objeto de revisión, entiende su rol y sus responsabilidades y recibe todo lo necesario para llevar a cabo la revisión.

Revisión Individual.

Cada revisor realiza una revisión individual para evaluar la calidad del producto del trabajo objeto de revisión e identificar anomalías, recomendaciones y preguntas aplicando una o varias técnicas de revisión (por ejemplo, revisión basada en lista de comprobación, revisión basada en escenarios). El estándar ISO/IEC 20246 profundiza en las diferentes técnicas de revisión. Los revisores registran todas las anomalías, recomendaciones y preguntas identificadas.

Comunicación y Análisis.

Dado que las anomalías identificadas durante una revisión no son necesariamente defectos, todas estas anomalías necesitan ser analizadas y discutidas. Para cada anomalía, debe tomarse una

decisión sobre su estado, propiedad y acciones requeridas. Esto suele hacerse en una reunión de revisión, durante la cual los participantes también deciden cuál es el nivel de calidad del producto del trabajo revisado y qué acciones de seguimiento se requieren. Puede ser necesaria una revisión de seguimiento para completar las acciones.

Corrección y Suministro de Información.

Por cada defecto, debe crearse un informe de defecto para poder hacer un seguimiento de las acciones correctivas. Una vez alcanzados los criterios de salida, el producto del trabajo puede ser aceptado. Se informa de los resultados de la revisión.

3.2.3 Roles y Responsabilidades en las Revisiones

Las revisiones implican a varios implicados, que pueden asumir varios roles. Los principales roles y sus responsabilidades son:

- Gestor - decide qué se va a revisar y aporta recursos, como personal y tiempo para la revisión
- Autor - crea y corrige el producto del trabajo objeto de revisión
- Moderador (también conocido como facilitador) - asegura el funcionamiento eficaz de las reuniones de revisión, incluyendo la mediación, la gestión del tiempo y un entorno de revisión seguro en el que todos puedan hablar libremente.
- Escriba (también conocido como grabador) - recopila las anomalías de los revisores y registra la información de la revisión, como las decisiones y las nuevas anomalías encontradas durante la reunión de revisión.
- Revisor - lleva a cabo las revisiones. Un revisor puede ser alguien que trabaje en el proyecto, un experto en la materia o cualquier otro implicado
- Líder de la revisión - asume la responsabilidad general de la revisión, como decidir quién participará y organizar cuándo y dónde tendrá lugar la revisión

Son posibles otros roles más detallados, como se describe en el estándar ISO/IEC 20246.

3.2.4 Tipos de Revisión

Existen muchos tipos de revisión, desde revisiones informales hasta revisiones formales. El nivel de formalidad requerido depende de factores como el CVDS que se siga, la madurez del proceso de desarrollo, la criticidad y complejidad del producto de trabajo que se revise, los requisitos legales o reglamentarios y la necesidad de un rastro de auditoría. Un mismo producto de trabajo puede ser revisado con distintos tipos de revisión, por ejemplo, primero una informal y después una más formal.

Seleccionar el tipo de revisión adecuado es clave para alcanzar los objetivos de revisión requeridos (véase la sección 3.2.5). La selección no sólo se basa en los objetivos, sino también en factores como las necesidades del proyecto, los recursos disponibles, el tipo de producto y los riesgos del trabajo, el dominio del negocio y la cultura de la empresa.

Algunos de los tipos de revisión más utilizados son:

- **Revisión Informal.**
  - Las revisiones informales no siguen un proceso definido y no requieren una salida formal documentada. El objetivo principal es detectar anomalías.
- **Revisión Guiada.**

- Una revisión guiada, dirigida por el autor, puede servir para muchos objetivos, como evaluar la calidad y generar confianza en el producto del trabajo, educar a los revisores, obtener consenso, generar nuevas ideas, motivar y capacitar a los autores para mejorar y detectar anomalías. Los revisores pueden realizar una revisión individual antes de la revisión guiada, pero no es un requisito.

- **Revisión Técnica.**
  - Una revisión técnica es realizada por revisores técnicamente cualificados y dirigida por un moderador. Los objetivos de una revisión técnica son obtener consenso y tomar decisiones sobre un problema técnico, pero también detectar anomalías, evaluar la calidad y generar confianza en el producto del trabajo, generar nuevas ideas y motivar y permitir a los autores mejorar.

- **Inspección.**
  - Como las inspecciones son el tipo de revisión más formal, siguen el proceso genérico completo (véase la sección 3.2.2). El objetivo principal es encontrar el máximo número de anomalías. Otros objetivos son evaluar la calidad, generar confianza en el producto del trabajo y motivar y capacitar a los autores para mejorar. Se recopilan métricas y se utilizan para mejorar el CVDS, incluido el proceso de inspección. En las inspecciones, el autor no puede actuar como revisor o escriba.

3.2.5 Factores de Éxito de las Revisiones

Hay diversos factores que determinan el éxito de las revisiones, entre los que se incluyen:

- La definición de objetivos claros y criterios de salida medibles. La evaluación de los participantes nunca debe ser un objetivo.
- Elegir el tipo de revisión adecuado para alcanzar los objetivos fijados y para adaptarse al tipo de producto de trabajo, a los participantes en la revisión, a las necesidades del proyecto y al contexto.
- Realizar las revisiones en pequeños fragmentos, para que los revisores no pierdan la concentración durante una revisión individual y/o la reunión de revisión (cuando se celebre).
- Proporcionar retroalimentación de las revisiones a los implicados y a los autores para que puedan mejorar el producto y sus actividades (véase la sección 3.2.1).
- Proporcionar tiempo suficiente a los participantes para prepararse para la revisión.
- Apoyo de la dirección al proceso de revisión.
- Hacer que las revisiones formen parte de la cultura de la organización, para promover el aprendizaje y la mejora del proceso.
- Proporcionar una formación adecuada a todos los participantes para que sepan cómo desempeñar su rol.
- Facilitar las reuniones.

4 Análisis y Diseño de la Prueba - 390 minutos

Duración: 390 minutos

Palabras Clave[5]

Español	Inglés
criterios de aceptación	acceptance criteria
desarrollo guiado por prueba de aceptación	acceptance test-driven development
técnica de prueba de caja negra	black-box test technique
análisis del valor frontera	boundary value analysis
cobertura de rama	branch coverage
prueba basada en lista de comprobación	checklist-based testing
enfoque de prueba basado en la colaboración	collaboration-based test approach
cobertura	coverage
elemento de cobertura	coverage item
prueba de tabla de decisión	decision table testing
partición de equivalencia	equivalence partitioning
predicción de errores	error guessing
técnica de prueba basada en la experiencia	experience-based test technique
prueba exploratoria	exploratory testing
prueba de transición de estado	state transition testing
cobertura de sentencia	statement coverage
técnica de prueba	test technique
técnica de prueba de caja blanca	white-box test technique

Objetivos de Aprendizaje para "Capítulo 4"

4.1 Introducción a las Técnicas de Prueba

FL-4.1.1	(K2)	Distinguir entre técnicas de prueba de caja negra, de caja blanca y basadas en la experiencia.

4.2 Técnicas de Prueba de Caja Negra

FL-4.2.1	(K3)	Utilizar partición de equivalencia para obtener casos de prueba.
FL-4.2.2	(K3)	Utilizar análisis del valor frontera para obtener casos de prueba.
FL-4.2.3	(K3)	Utilizar prueba de tabla de decisión para obtener casos de prueba.
FL-4.2.4	(K3)	Utilizar prueba de transición de estado para obtener casos de prueba.

[5] Las palabras clave se encuentran ordenadas por orden alfabético de los términos en inglés.

4.3 Técnicas de Prueba de Caja Blanca		
FL-4.3.1	(K2)	Explicar la prueba de sentencia.
FL-4.3.2	(K2)	Explicar la prueba de rama.
FL-4.3.3	(K2)	Explicar el valor de la prueba de caja blanca.
4.4 Técnicas de Prueba Basadas en la Experiencia		
FL-4.4.1	(K2)	Explicar la predicción de errores.
FL-4.4.2	(K2)	Explicar la prueba exploratoria.
FL-4.4.3	(K2)	Explicar la prueba basada en lista de comprobación.
4.5. Enfoques de Prueba Basados en la Colaboración		
FL-4.5.1	(K2)	Explicar cómo escribir historias de usuario en colaboración con desarrolladores y representantes de negocio.
FL-4.5.2	(K2)	Clasificar las diferentes opciones para escribir criterios de aceptación.
FL-4.5.3	(K3)	Utilizar el desarrollo guiado por prueba de aceptación (DGPA) para obtener casos de prueba.

4.1 Introducción a las Técnicas de Prueba

Las técnicas de prueba ayudan al probador en el análisis de prueba (qué probar) y en el diseño de prueba (cómo probar). Las técnicas de prueba ayudan a desarrollar un conjunto relativamente pequeño, pero suficiente, de casos de prueba de forma sistemática. Las técnicas de prueba también ayudan al probador a definir las condiciones de la prueba, identificar los elementos de cobertura e identificar los datos de prueba durante el análisis y el diseño de la prueba. Se puede encontrar más información sobre las técnicas de prueba y sus correspondientes medidas en el estándar ISO/IEC/IEEE 29119-4 y en (Beizer 1990, Craig 2002, Copeland 2004, Koomen 2006, Jorgensen 2014, Ammann 2016, Forgács 2019).

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

Técnicas de Prueba de Caja Negra

Las técnicas de prueba de caja negra (también conocidas como técnicas basadas en la especificación) se basan en un análisis del comportamiento especificado del objeto de prueba sin hacer referencia a su estructura interna. Por lo tanto, los casos de prueba son independientes de cómo se implemente el software. En consecuencia, si la implementación cambia, pero el comportamiento requerido sigue siendo el mismo, los casos de prueba siguen siendo útiles.

Técnicas de Prueba de Caja Blanca

Las técnicas de prueba de caja blanca (también conocidas como técnicas basadas en la estructura) se basan en un análisis de la estructura interna y el procesamiento del objeto de prueba. Como los casos de prueba dependen de cómo esté diseñado el software, sólo pueden crearse después del diseño o la implementación del objeto de prueba.

Técnicas de Prueba Basadas en la Experiencia

Las técnicas de prueba basadas en la experiencia utilizan eficazmente los conocimientos y la experiencia de los probadores para el diseño y la implementación de los casos de prueba. La efectividad de estas técnicas depende en gran medida de las competencias del probador. Las técnicas de prueba basadas en la experiencia pueden detectar defectos que no se detectan con las técnicas de prueba de caja blanca y caja negra. Por lo tanto, las técnicas de prueba basadas en la experiencia son complementarias a las técnicas de prueba de caja blanca y de caja negra.

4.2 Técnicas de Prueba de Caja Negra

Las técnicas de prueba de caja negra utilizadas de forma habitual y que se analizan en las secciones siguientes son:

- Partición de equivalencia
- Análisis del valor frontera
- Prueba de tabla de decisión
- Prueba de transición de estado

4.2.1 Partición de Equivalencia

La Partición de Equivalencia (PE) divide los datos en particiones (conocidas como particiones de equivalencia) basándose en la expectativa de que todos los elementos de una partición determinada van a ser procesados de la misma manera por el objeto de prueba. La teoría que subyace a esta técnica es que, si un caso de prueba, que prueba un valor de una partición de equivalencia, detecta un defecto, este defecto también debería ser detectado por los casos de prueba que prueban cualquier otro valor de la misma partición. Por lo tanto, basta con una prueba para cada partición.

Las particiones de equivalencia pueden identificarse para cualquier elemento de dato relacionado con el objeto de prueba, incluyendo entradas | salidas, elementos de configuración, valores internos, valores relacionados con el tiempo y parámetros de interfaz. Las particiones pueden ser continuas o discretas, ordenadas o desordenadas, finitas o infinitas. Las particiones no deben solaparse y deben ser conjuntos no vacíos.

Para objetos de prueba simples, la PE puede ser fácil, pero en la práctica, entender cómo tratará el objeto de prueba los distintos valores suele ser complicado. Por lo tanto, la partición debe hacerse con cuidado.

Una partición que contenga valores válidos se denomina partición válida. Una partición que contiene valores no válidos se denomina partición no válida. Las definiciones de valores válidos e inválidos pueden variar entre equipos y organizaciones. Por ejemplo, los valores válidos pueden interpretarse como aquellos que deben ser procesados por el objeto de prueba o como aquellos para los que la especificación define su procesamiento. Los valores no válidos pueden interpretarse como aquellos que deben ser ignorados o rechazados por el objeto de prueba o como aquellos para los que no se define su procesamiento en la especificación del objeto de prueba.

Una partición que contenga valores válidos se denomina partición válida. Una partición que contiene valores no válidos se denomina partición no válida. Las definiciones de valores válidos e inválidos pueden variar entre equipos y organizaciones. Por ejemplo, los valores válidos pueden interpretarse como aquellos que deben ser procesados por el objeto de prueba o como aquellos para los que la especificación define su procesamiento. Los valores no válidos pueden interpretarse como aquellos que deben ser ignorados o rechazados por el objeto de prueba o como aquellos para los que no se define su procesamiento en la especificación del objeto de prueba.

Muchos objetos de prueba incluyen múltiples conjuntos de particiones (por ejemplo, objetos de prueba con más de un parámetro de entrada), lo que significa que un caso de prueba cubrirá particiones de diferentes conjuntos de particiones. El criterio de cobertura más sencillo en el caso de conjuntos múltiples de particiones se denomina cobertura de Cada Elección (Ammann 2016). La cobertura de Cada Elección exige que los casos de prueba utilicen cada partición de cada conjunto de particiones al menos una vez. La cobertura Cada Elección no tiene en cuenta las combinaciones de particiones.

4.2.2 Análisis del Valor Frontera

El Análisis del Valor Frontera (AVF) es una técnica basada en practicar las fronteras de las particiones de equivalencia. Por lo tanto, el AVF sólo puede utilizarse para particiones ordenadas. Los valores mínimo y máximo de una partición son sus valores frontera. En el caso de AVF, si dos elementos pertenecen a la misma partición, todos los elementos entre ellos deben pertenecer también a esa partición.

El AVF se concentra en los valores frontera de las particiones porque es más probable que los desarrolladores cometan errores con estos valores frontera. Los defectos típicos que encuentra el BVA se localizan cuando las fronteras implementadas se colocan erróneamente en posiciones superiores o inferiores a las previstas o se omiten por completo.

Este programa de estudio abarca dos versiones del AVF: AVF de 2 valores y AVF de 3 valores. Se diferencian en cuanto a los elementos de cobertura por frontera que necesitan ser practicados para lograr una cobertura del 100%.

En el AVF de 2 valores (Craig 2002, Myers 2011), para cada valor frontera hay dos elementos de cobertura: este valor frontera y su vecino más cercano perteneciente a la partición adyacente. Para lograr una cobertura del 100% con BVA de 2 valores, los casos de prueba deben utilizar todos los elementos de cobertura, es decir, todos los valores frontera identificados. La cobertura se mide como el número de valores frontera que se han practicado, dividido por el número total de valores frontera identificados, y se expresa en porcentaje.

En el AVF de 3 valores (Koomen 2006, O'Regan 2019), para cada valor frontera hay tres elementos de cobertura: este valor frontera y sus dos vecinos. Por lo tanto, en el AVF de 3 valores algunos de los elementos de cobertura pueden no ser valores frontera. Para lograr una cobertura del 100% con BVA de 3 valores, los casos de prueba deben practicar todos los elementos de cobertura, es decir, los valores frontera identificados y sus vecinos. La cobertura se mide como el número de valores frontera y sus vecinos practicados, dividido por el número total de valores frontera identificados y sus vecinos, y se expresa en porcentaje.

El AVF de 3 valores es más riguroso que el AVF de 2 valores, ya que puede detectar defectos pasados por alto por el AVF de 2 valores. Por ejemplo, si la decisión "si (x = 10) ..." se implementa incorrectamente como "si (x = 10) ...", ningún dato de prueba derivado del AVF de 2 valores (x = 10, x = 11) puede detectar el defecto. Sin embargo, es probable que x = 9, derivado del AVF de 3 valores, lo detecte.

4.2.3 Prueba de Tabla de Decisión

Las tablas de decisión se utilizan para probar la implementación de los requisitos de sistemas que especifican cómo diferentes combinaciones de condiciones dan lugar a diferentes resultados. Las tablas de decisión son una forma efectiva de registrar lógicas complejas, como las reglas de negocio.

Cuando se crean tablas de decisión, se definen las condiciones y las acciones resultantes del sistema. Éstas forman las filas de la tabla. Cada columna corresponde a una regla de decisión que define una combinación única de condiciones, junto con las acciones asociadas. En las tablas de decisión de entrada limitada, todos los valores de las condiciones y acciones (excepto los irrelevantes o inviables; véase más adelante) se muestran como valores booleanos (verdadero o falso). Alternativamente, en las tablas de decisión de entrada ampliada algunas o todas las condiciones y acciones también pueden adoptar valores múltiples (por ejemplo, rangos de números, particiones de equivalencia, valores discretos).

La notación para las condiciones es la siguiente: "V" (verdadero) significa que se cumple la condición. "F" (falso) significa que la condición no se satisface. "-" significa que el valor de la condición es irrelevante para el resultado de la acción. "N/A" significa que la condición es inviable para una regla determinada. Para las acciones: "X" significa que la acción debe producirse. En blanco significa que la acción no debe ocurrir. También se pueden utilizar otras notaciones.

Una tabla de decisión completa tiene suficientes columnas para cubrir todas las combinaciones de condiciones. La tabla puede simplificarse eliminando las columnas que contengan combinaciones de condiciones inviables. La tabla también se puede minimizar fusionando en una sola columna las columnas en las que algunas condiciones no afectan al resultado. Los algoritmos de minimización de tablas de decisión están fuera del alcance de este programa de estudio.

En la prueba de tabla de decisión, los elementos de cobertura son las columnas que contienen combinaciones de condiciones factibles. Para lograr una cobertura del 100% con esta técnica, los casos de prueba deben practicar todas estas columnas. La cobertura se mide como el número de columnas practicadas, dividido por el número total de columnas factibles, y se expresa en porcentaje.

El punto fuerte de la prueba de tabla de decisión es que proporciona un enfoque sistemático para identificar todas las combinaciones de condiciones, algunas de las cuales podrían pasarse por alto de otro modo. También ayuda a encontrar lagunas o contradicciones en los requisitos. Si hay muchas condiciones, practicar todas las reglas de decisión puede llevar mucho tiempo, ya que el número de reglas crece exponencialmente con el número de condiciones. En tal caso, para reducir el número de reglas que necesitan ser practicadas, puede utilizarse una tabla de decisión minimizada o un enfoque basado en el riesgo.

4.2.4 Prueba de Transición de Estado

Un diagrama de transición de estados modela el comportamiento de un sistema mostrando sus posibles estados y las transiciones de estado válidas. Una transición es iniciada por un evento, que puede ser calificado adicionalmente por una condición de guarda. Se supone que las transiciones son instantáneas y, en ocasiones, pueden dar lugar a que el software pase a la acción. La sintaxis común de etiquetado de transiciones es la siguiente: "evento [condición de guardia] / acción]". Las condiciones de guarda y las acciones pueden omitirse si no existen o son irrelevantes para el probador.

Una tabla de estados es un modelo equivalente a un diagrama de transición de estado. Sus filas representan estados y sus columnas eventos (junto con las condiciones de guarda si existen). Las entradas de la tabla (celdas) representan transiciones, y contienen el estado objetivo, así como las acciones resultantes, si están definidas. A diferencia del diagrama de transición de estados, la tabla de estados muestra explícitamente las transiciones no válidas, que se representan mediante celdas vacías.

Un caso de prueba basado en un diagrama de transición de estado o en una tabla de estados suele representarse como una secuencia de eventos, que da lugar a una secuencia de cambios de estado (y acciones, si se necesitan). Un caso de prueba puede cubrir, y normalmente cubrirá, varias transiciones entre estados.

Existen muchos criterios de cobertura para las pruebas de transición de estado. Este programa de estudio analiza tres de ellos.

En la cobertura de todos los estados, los elementos de cobertura son los estados. Para lograr una cobertura del 100% de todos los estados, los casos de prueba deben asegurar que se visitan todos los estados. La cobertura se mide como el número de estados visitados dividido por el número total de estados, y se expresa en porcentaje.

En la cobertura de transiciones válidas (también denominada cobertura de conmutador 0), los elementos de cobertura son transiciones válidas únicas. Para lograr una cobertura de transiciones válidas del 100%, los casos de prueba deben practicar todas las transiciones válidas. La cobertura se mide como el número de transiciones válidas practicadas dividido por el número total de transiciones válidas, y se expresa en porcentaje.

En la cobertura de todas las transiciones, los elementos de cobertura son todas las transiciones que aparecen en una tabla de estados. Para lograr una cobertura del 100% de todas las transiciones, los casos de prueba deben practicar todas las transiciones válidas e intentar ejecutar las transiciones no válidas.

Probar sólo una transición no válida en un único caso de prueba ayuda a evitar el enmascaramiento de defecto, es decir, una situación en la que un defecto impide la detección de otro. La cobertura se mide como el número de transiciones válidas e inválidas practicadas o intentadas por los casos de prueba ejecutados, dividido por el número total de transiciones válidas e inválidas, y se expresa en porcentaje.

La cobertura de todos los estados es más débil que la cobertura de las transiciones válidas, porque normalmente puede lograrse sin practicar todas las transiciones. La cobertura de transiciones válidas es el criterio de cobertura más utilizado. Lograr una cobertura completa de transiciones válidas garantiza una cobertura completa de todos los estados. Lograr la cobertura completa de todas las transiciones garantiza tanto la cobertura completa de todos los estados como la cobertura completa de las transiciones válidas y debería ser un requisito mínimo para el software de misión y seguridad física críticas.

4.3 Técnicas de Prueba de Caja Blanca

Debido a su popularidad y sencillez, esta sección se concentra en dos técnicas de prueba de caja blanca relacionadas con el código:

- Prueba de sentencia
- Pruebas de rama

Existen técnicas más rigurosas que se utilizan en algunos entornos de seguridad crítica, misión crítica o alta integridad para lograr una cobertura de código más profunda. También hay técnicas de prueba de caja blanca que se utilizan en niveles de prueba superiores (por ejemplo, prueba de API), o que utilizan una cobertura no relacionada con el código (por ejemplo, la cobertura de neuronas en la prueba de redes neuronales). Estas técnicas no se tratan en este programa de estudio.

4.3.1 Prueba de Sentencia y Cobertura de Sentencia

En la prueba de sentencia, los elementos de cobertura son sentencias ejecutables. El objetivo es diseñar casos de prueba que practiquen sentencias en el código hasta alcanzar un nivel aceptable de cobertura. La cobertura se mide como el número de sentencias practicadas por los casos de prueba dividido por el número total de sentencias ejecutables del código, y se expresa en porcentaje.

Cuando se alcanza el 100% de cobertura de sentencias, se asegura que todas las sentencias ejecutables del código han sido practicadas al menos una vez. En concreto, esto significa que se ejecutará cada sentencia con un defecto, lo que puede provocar un fallo que demuestre la presencia del defecto. Sin embargo, practicar una sentencia con un caso de prueba no detectará defectos en todos los casos. Por ejemplo, puede que no detecte defectos que dependen de los datos (por ejemplo, una división por cero que sólo falla cuando el denominador se pone a cero). Además, una cobertura de sentencia del 100% no asegura que se haya practicado toda la lógica de decisión ya que, por ejemplo, puede que no se practiquen todas las ramas (véase el capítulo 4.3.2) del código.

4.3.2 Prueba de Rama y Cobertura de Rama

Una rama es una transferencia de control entre dos nodos del grafo del flujo de control, que muestra las posibles secuencias en las que se ejecutan las sentencias del código fuente en el objeto de prueba. Cada transferencia de control puede ser incondicional (es decir, código en línea recta) o condicional (es decir, un resultado de la decisión).

En la prueba de rama, los elementos de cobertura son las ramas y el objetivo es diseñar casos de prueba que permitan practicar las ramas del código hasta alcanzar un nivel de cobertura aceptable. La cobertura se mide como el número de ramas practicadas por los casos de prueba dividido por el número total de ramas, y se expresa en porcentaje.

Cuando se alcanza el 100% de cobertura de rama, todas las ramas del código, incondicionales y condicionales, son practicadas por los casos de prueba. Las ramas condicionales suelen corresponder a un resultado verdadero o falso de una decisión "if...then", un resultado de una sentencia switch/case o una decisión de salir o continuar en un bucle. Sin embargo, practicar una rama con un caso de prueba no detectará defectos en todos los casos. Por ejemplo, puede que no detecte defectos que requieran la ejecución de un camino específico en un código.

La cobertura de rama subsume la cobertura de sentencia. Esto significa que cualquier conjunto de casos de prueba que logre una cobertura de rama del 100% también logra una cobertura de sentencia del 100% (pero no viceversa).

4.3.3 El valor de la Prueba de Caja Blanca

Un punto fuerte fundamental que comparten todas las técnicas de caja blanca es que durante las pruebas se tiene en cuenta toda la implementación del software, lo que facilita la detección de defectos incluso cuando la especificación del software es vaga, obsoleta o incompleta. Un punto débil correspondiente es

que si el software no implementa uno o más requisitos, la prueba de caja blanca puede no detectar los defectos de omisión resultantes (Watson 1996).

Las técnicas de caja blanca pueden utilizarse en la prueba estática (por ejemplo, durante la realización de simulacros de código). Son muy adecuadas para revisar código que aún no está listo para su ejecución (Hetzel 1988), así como pseudocódigo y otra lógica de alto nivel o descendente que pueda modelarse con un grafo del flujo de control.

Realizar sólo la prueba de caja negra no proporciona una medida de la cobertura de código real. Las medidas de cobertura de caja blanca proporcionan una medición objetiva de la cobertura y aportan la información necesaria para permitir que se generen pruebas adicionales que aumenten esta cobertura y, posteriormente, aumenten la confianza en el código.

4.4 Técnicas de Prueba Basadas en la Experiencia

Las técnicas de prueba basadas en la experiencia utilizadas de forma habitual y que se analizan en las secciones siguientes son:

- Predicción de errores.
- Prueba exploratoria.
- Prueba basada en lista de comprobación.

4.4.1 Predicción de Errores

La predicción de errores es una técnica utilizada para anticipar la aparición de errores, defectos y fallos, basada en los conocimientos del probador, entre los que se incluyen:

- Cómo ha funcionado la aplicación en el pasado.
- Los tipos de errores que suelen cometer los desarrolladores y los tipos de defectos que resultan de estos errores.
- Los tipos de fallos que se han producido en otras aplicaciones similares.

En general, los errores, defectos y fallos pueden estar relacionados con: la entrada (por ejemplo, no se acepta la entrada correcta, parámetros erróneos o ausentes), la salida (por ejemplo, formato incorrecto, resultado erróneo), la lógica (por ejemplo, casos ausentes, operador erróneo), el cálculo (por ejemplo, operando incorrecto, cálculo erróneo), las interfaces (por ejemplo, desajuste de parámetros, tipos incompatibles) o los datos (por ejemplo, inicialización incorrecta, tipo erróneo).

Los ataques de defecto son un enfoque metódico de la implementación de la predicción de errores. Esta técnica requiere que el probador cree o adquiera una lista de posibles errores, defectos y fallos, y que diseñe pruebas que identifiquen los defectos asociados a los errores, expongan los defectos o provoquen los fallos. Estas listas pueden construirse basándose en la experiencia, en datos sobre defectos y fallos o en el conocimiento común sobre por qué falla el software.

Para más información sobre la predicción de errores y los ataques de defecto, véase (Whittaker 2002, Whittaker 2003, Andrews 2006).

4.4.2 Prueba Exploratoria

En la prueba exploratoria, las pruebas se diseñan, ejecutan y evalúan simultáneamente mientras el probador aprende sobre el objeto de prueba. La prueba se utiliza para aprender más sobre el objeto de prueba, para explorarlo más profundamente con pruebas concentradas y para crear pruebas para áreas no probadas.

A veces, la prueba exploratoria se lleva a cabo en la forma de prueba basada en la sesión para estructurar la prueba. En un enfoque basado en la sesión, la prueba exploratoria se realiza dentro de un marco temporal definido. El probador utiliza un contrato de prueba que contiene los objetivos de prueba para guiar la prueba. La sesión de prueba suele ir seguida de una recapitulación que implica un debate entre el probador y los implicados en los resultados de prueba de la sesión de prueba. En este enfoque, los objetivos de prueba pueden tratarse como condiciones de prueba de alto nivel. Los elementos de cobertura se identifican y se practican durante la sesión de prueba. El probador puede utilizar hojas de sesión de prueba para documentar los pasos seguidos y los descubrimientos realizados.

La prueba exploratoria resulta útil cuando las especificaciones son escasas o inadecuadas o cuando existe una presión de tiempo importante para la prueba. La prueba exploratoria también es útil para complementar otras técnicas de prueba más formales. La prueba exploratoria será más eficaz si el probador tiene

experiencia, conoce el dominio y posee un alto grado de competencias esenciales, como capacidad de análisis, curiosidad y creatividad (véase el apartado 1.5.1).

La prueba exploratoria puede incorporar el uso de otras técnicas de prueba (por ejemplo, la partición de equivalencia). Puede encontrar más información sobre las pruebas exploratorias en (Kaner 1999, Whittaker 2009, Hendrickson 2013).

4.4.3 Prueba basada en Lista de Comprobación

En la prueba basada en lista de comprobación, un probador diseña, implementa y ejecuta pruebas para cubrir las condiciones de prueba de una lista de comprobación. Las listas de comprobación pueden construirse basándose en la experiencia, en el conocimiento de lo que es importante para el usuario o en la comprensión de por qué y cómo falla el software. Las listas de comprobación no deben contener elementos que puedan comprobarse automáticamente, elementos más adecuados como criterios de entrada/salida o elementos demasiado generales (Brykczynski 1999).

Los elementos de la lista de comprobación suelen formularse en forma de pregunta. Debe ser posible comprobar cada elemento por separado y directamente. Estos elementos pueden referirse a requisitos, propiedades de la interfaz gráfica, características de calidad u otras formas de condiciones de prueba. Se pueden crear listas de comprobación para apoyar diversos tipos de pruebas, incluidas las pruebas funcionales y no funcionales (por ejemplo, 10 heurísticas para pruebas de usabilidad (Nielsen 1994)).

Algunas entradas de la lista de comprobación pueden ir perdiendo efectividad con el tiempo porque los desarrolladores aprenderán a evitar cometer los mismos errores. También puede necesitar que se añadan nuevas entradas para reflejar defectos de alta severidad recién encontrados. Por lo tanto, las listas de comprobación deben actualizarse periódicamente en función del análisis de los defectos. Sin embargo, hay que tener cuidado para evitar que la lista de comprobación se haga excesivamente larga (Gawande 2009).

A falta de casos de comprobación detallados, la prueba basada en listas de comprobación puede proporcionar directrices y cierto grado de consistencia para la prueba. Si las listas de comprobación son 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.

4.5 Enfoques de Prueba Basados en la Colaboración

Cada una de las técnicas mencionadas anteriormente (véanse las secciones 4.2, 4.3 y 4.4) tiene un objetivo particular con respecto a la detección de defectos. Los enfoques basados en la colaboración, en cambio, se concentran también en evitar defectos mediante la colaboración y la comunicación.

4.5.1 Redacción Colaborativa de Historias de Usuario

Una historia de usuario representa una prestación que será de valor para un usuario o comprador de un sistema o software. Las historias de usuario tienen tres aspectos críticos (Jeffries 2000), denominados en conjunto las "3 C"[6] (en referencia a las iniciales de los términos en inglés: **C**ard, **C**onversation, **C**onfirmation).

Inglés	Español	
Card →	**C**uartilla -	el medio que describe una historia de usuario (por ejemplo, una ficha, una entrada en un tablón electrónico)
Conversation →	**C**onversación -	explica cómo se utilizará el software (puede ser documentada o verbal)
Confirmation →	**C**onfirmación -	los criterios de aceptación (véase la sección 4.5.2)

El formato más habitual de una historia de usuario es "Como [rol], quiero que se cumpla [objetivo], de modo que pueda [valor de negocio resultante para el rol]", seguido de los criterios de aceptación.

La autoría colaborativa de la historia de usuario puede utilizar técnicas como la tormenta de ideas y los mapas mentales. La colaboración permite al equipo obtener una visión compartida de lo que debe entregarse, teniendo en cuenta tres perspectivas: el negocio, el desarrollo y la prueba.

Las buenas historias de usuario deben ser: Independientes, Negociables, Valiosas, Estimables, Pequeñas y Comprobables (INVEST, en referencia a las iniciales de los términos en inglés: **I**ndependent**, N**egotiable**, V**aluable**, E**stimable**, S**mall **and T**estable). Si un implicado no sabe cómo probar una historia de usuario, esto puede indicar que la historia de usuario no es lo suficientemente clara, o que no refleja algo valioso para él, o que simplemente necesita ayuda para probarla (Wake 2003).

Español		Inglés
Independiente	**I**	**I**ndependent
Negociable	**N**	**N**egotiable
Valioso/a	**V**	**V**aluable
Estimable	**E**	**E**stimable
Pequeño/a	**S**	**S**mall
Capacidad de Prueba	**T**	**T**estable

[6] En este caso y contexto, “cuartilla” es la traducción del término “card”. La traducción correcta sería “tarjeta” pero se optó por cuartilla para conservar el mnemónico “3C”. Una cuartilla es una hoja de papel de tamaño A5 (148mm x 210 mm).

4.5.2 Criterios de Aceptación

Los criterios de aceptación de una historia de usuario son las condiciones que debe cumplir una implementación de la historia de usuario para ser aceptada por los implicados. Desde esta perspectiva, los criterios de aceptación pueden verse como las condiciones de prueba que deben ser practicadas por las pruebas. Los criterios de aceptación suelen ser un resultado de la Conversación (véase la sección 4.5.1).

Los criterios de aceptación se utilizan para:

- Definir el alcance de la historia de usuario.
- Alcanzar un consenso entre los implicados.
- Describir escenarios positivos y negativos.
- Servir de base para la prueba de aceptación de usuario (véase la sección 4.5.3).
- Permitir una planificación y estimación precisas.

Hay varias formas de redactar los criterios de aceptación de una historia de usuario. Los dos formatos más comunes son:

- Orientado al escenario (por ejemplo, el formato Dado/Cuando/Entonces (“Given/When/Then”) utilizado en desarrollo guiado por el comportamiento (DGC), véase la sección 2.1.3)
- Orientado a reglas (por ejemplo, lista de verificación con viñetas, o forma tabulada de mapeo entrada-salida)

La mayoría de los criterios de aceptación pueden documentarse en uno de estos dos formatos. Sin embargo, el equipo puede utilizar otro formato personalizado, siempre que los criterios de aceptación estén bien definidos y sean inequívocos.

4.5.3 Desarrollo Guiado por Prueba de Aceptación (DGPA)

DGPA es un enfoque de tipo probar primero (véase la sección 2.1.3). Los casos de prueba se crean antes de la implementación de la historia de usuario. Los casos de prueba son creados por miembros del equipo con diferentes perspectivas, por ejemplo, clientes, desarrolladores y probadores (Adzic 2009). Los casos de prueba pueden ejecutarse de forma manual o automatizada.

El primer paso es un taller de especificación en el que los miembros del equipo analizan, debaten y redactan la historia de usuario y (si aún no están definidos) sus criterios de aceptación. Durante este proceso se resuelven las incompletitudes, ambigüedades o defectos de la historia de usuario. El siguiente paso consiste en crear los casos de prueba. Esto puede hacerlo el equipo en su conjunto o el probador individualmente. Los casos de prueba se basan en los criterios de aceptación y pueden verse como ejemplos de cómo funciona el software. Esto ayudará al equipo a implementar correctamente la historia de usuario.

Dado que ejemplos y pruebas son lo mismo, estos términos suelen utilizarse indistintamente. Durante el diseño de prueba pueden aplicarse las técnicas de prueba descritas en las secciones 4.2, 4.3 y 4.4.

Normalmente, los primeros casos de prueba son positivos, confirman el comportamiento correcto sin excepciones ni condiciones de error y comprenden la secuencia de actividades que se ejecutan si todo va como se espera. Una vez realizados los casos de prueba positivos, el equipo debe realizar pruebas negativas. Por último, el equipo debe cubrir también las características de calidad no funcionales (por ejemplo, la eficiencia de desempeño, la usabilidad). Los casos de prueba deben expresarse de forma

comprensible para los implicados. Normalmente, los casos de prueba contienen frases en lenguaje natural que incluyen las precondiciones necesarias (si las hay), las entradas y las postcondiciones.

Los casos de prueba deben cubrir todas las características de la historia de usuario y no deben ir más allá de la historia. Sin embargo, los criterios de aceptación pueden detallar algunos de los problemas descritos en la historia de usuario. Además, no debe haber dos casos de prueba que describan las mismas características de la historia de usuario.

Cuando se capturan en un formato compatible con un marco de automatización de la prueba, los desarrolladores pueden automatizar los casos de prueba escribiendo el código de apoyo a medida que implementan la prestación descrita por una historia de usuario. Las pruebas de aceptación se convierten entonces en requisitos ejecutables.

5 Gestión de las Actividades de Prueba - 335 minutos

Duración: 335 minutos

Palabras Clave[7]

Español	Inglés
gestión de defectos	defect management
informe de defecto	defect report
criterios de entrada	entry criteria
criterios de salida	exit criteria
riesgo de producto	product risk
riesgo de proyecto	project risk
riesgo	risk
análisis del riesgo	risk analysis
evaluación del riesgo	risk assessment
control del riesgo	risk control
identificación del riesgo	risk identification
nivel de riesgo	risk level
gestión del riesgo	risk management
mitigación del riesgo	risk mitigation
monitorización del riesgo	risk monitoring
pruebas basadas en el riesgo	risk-based testing
enfoque de prueba	test approach
informe de compleción de la prueba	test completion report
control de la prueba	test control
monitorización de la prueba	test monitoring
plan de prueba	test plan
planificación de prueba	test planning
informe del avance de la prueba	test progress report
pirámide de prueba	test pyramid
cuadrantes de prueba	testing quadrants

[7] Las palabras clave se encuentran ordenadas por orden alfabético de los términos en inglés.

Objetivos de Aprendizaje para "Capítulo 5"		
5.1 Planificación de la Prueba		
FL-5.1.1	(K2)	Dar ejemplos del propósito y el contenido de un plan de prueba.
FL-5.1.2	(K1)	Reconocer cómo un probador añade valor a la planificación de la iteración y de la entrega.
FL-5.1.3	(K2)	Comparar y contrastar los criterios de entrada y los criterios de salida.
FL-5.1.4	(K3)	Utilizar técnicas de estimación para calcular el esfuerzo de prueba necesario.
FL-5.1.5	(K3)	Aplicar la priorización de casos de prueba.
FL-5.1.6	(K1)	Recordar los conceptos de la pirámide de prueba.
FL-5.1.7	(K2)	Resumir los cuadrantes de prueba y sus relaciones con los niveles de prueba y los tipos de prueba.
5.2 Gestión del Riesgo		
FL-5.2.1	(K1)	Identificar el nivel de riesgo utilizando la probabilidad del riesgo y el impacto del riesgo.
FL-5.2.2	(K2)	Distinguir entre riesgos de proyecto y riesgos de producto.
FL-5.2.3	(K2)	Explicar cómo el análisis del riesgo de producto puede influir en la minuciosidad y el alcance de las pruebas.
FL-5.2.4	(K2)	Explique qué medidas pueden tomarse en respuesta a los riesgos de producto analizados.
5.3 Monitorización de la Prueba, Control de la Prueba y Compleción de la Prueba		
FL-5.3.1	(K1)	Recordar las métricas utilizadas para probar.
FL-5.3.2	(K2)	Resumir los propósitos, el contenido y las audiencias de los informes de prueba.
FL-5.3.3	(K2)	Dar ejemplos de cómo comunicar el estado de la prueba.
5.4 Gestión de la Configuración		
FL-5.4.1	(K2)	Resumir cómo la gestión de la configuración apoya la prueba.
5.5. Gestión de Defectos		
FL-5.5.1	(K3)	Preparar un informe de defecto.

5.1 Planificación de la Prueba

5.1.1 Propósito y Contenido de un Plan de Prueba

Un plan de prueba describe los objetivos, los recursos y los procesos de un proyecto de prueba. Un plan de pruebas:

- Documenta los medios y el calendario para alcanzar los objetivos de prueba.
- Ayuda a asegurar que las actividades de prueba realizadas cumplirán los criterios establecidos.
- Sirve como medio de comunicación con los miembros del equipo y otros implicados.
- Demuestra que las pruebas se ajustarán a la política de prueba y la estrategia de prueba existentes (o explica por qué las pruebas se desviarán de ellas).

La planificación de la prueba guía el pensamiento de los probadores y les obliga a enfrentarse a los retos futuros relacionados con los riesgos, los calendarios, las personas, las herramientas, los costes, el esfuerzo, etc. El proceso de elaboración de un plan de prueba es una forma útil de reflexionar sobre los esfuerzos necesarios para alcanzar los objetivos del proyecto de prueba.

El contenido habitual de un plan de prueba incluye:

- Contexto de la prueba (por ejemplo, alcance, objetivos de la prueba, restricciones, base de prueba).
- Supuestos y restricciones del proyecto de prueba.
- Implicados (por ejemplo, roles, responsabilidades, relevancia para las pruebas, necesidades de contratación y formación).
- Comunicación (por ejemplo, formas y frecuencia de la comunicación, plantillas de documentación).
- Registro de riesgos (por ejemplo, riesgos de producto, riesgos de proyecto).
- Enfoque de prueba (por ejemplo, niveles de prueba, tipos de prueba, técnicas de prueba, entregables de prueba, criterios de entrada y criterios de salida, independencia de prueba, métricas que deben recopilarse, requisitos de datos de prueba, requisitos del entorno de prueba, desviaciones de la política de prueba y la estrategia de prueba de la organización).
- Presupuesto y calendario.

Se pueden encontrar más detalles sobre el plan de pruebas y su contenido en el estándar ISO/IEC/IEEE 29119-3.

5.1.2 Contribución del Probador a la planificación de la Iteración y de la Entrega

En los CVDS iterativos, suelen producirse dos tipos de planificación: la planificación de la entrega y la planificación de la iteración.

La planificación de la entrega se anticipa al lanzamiento de un producto, define y redefine la lista de trabajo acumulado del producto y puede implicar refinar las historias de usuario más grandes en un conjunto de historias de usuario más pequeñas. También sirve de base para el enfoque de prueba y el plan de prueba de todas las iteraciones. Los encargados de la planificación de la entrega participan en la redacción de historias de usuario y criterios de aceptación comprobables (véase el apartado 4.5), participan en los análisis de riesgos de calidad y del proyecto (véase el apartado 5.2), estiman el esfuerzo de prueba

asociado a las historias de usuario (véase el apartado 5.1.4), determinan el enfoque de la prueba y planifican la prueba de la entrega.

La planificación de la iteración mira hacia el final de una única iteración y se ocupa de la lista de trabajo acumulado de la iteración. Los probadores implicados en la planificación de la iteración participan en el análisis detallado del riesgo de las historias de usuario, determinan la capacidad de ser probadas de las historias de usuario, desglosan las historias de usuario en tareas (en particular, tareas de prueba), estiman el esfuerzo de prueba para todas las tareas de prueba e identifican y perfeccionan los aspectos funcionales y no funcionales del objeto de prueba.

5.1.3 Criterios de Entrada y Criterios de Salida

Los criterios de entrada definen las precondiciones para emprender una actividad determinada. Si no se cumplen los criterios de entrada, es probable que la actividad resulte más difícil, lenta, costosa y arriesgada. Los criterios de salida definen lo que debe lograrse para declarar completada una actividad. Los criterios de entrada y los criterios de salida deben definirse para cada nivel de prueba, y diferirán en función de los objetivos de prueba.

Los criterios de entrada típicos incluyen: disponibilidad de recursos (por ejemplo, personas, herramientas, entornos, datos de prueba, presupuesto, tiempo), disponibilidad de material de prueba (por ejemplo, base de prueba, requisitos comprobables, historias de usuario, casos de prueba) y nivel de calidad inicial de un objeto de prueba (por ejemplo, todas las pruebas de humo han pasado).

Los criterios de salida típicos incluyen: medidas de completitud (p. ej., nivel de cobertura alcanzado, número de defectos sin resolver, densidad de defectos, número de casos de prueba fallidos) y criterios de compleción (p. ej., se han ejecutado las pruebas planificadas, se ha realizado la prueba estática, se han notificado todos los defectos encontrados, se han automatizado todas las pruebas de regresión).

Quedarse sin tiempo o sin presupuesto también pueden considerarse criterios de salida válidos. Incluso sin que se cumplan otros criterios de salida, puede ser aceptable finalizar las pruebas en tales circunstancias, si los implicados han revisado y aceptado el riesgo de salir a producción sin más prueba.

En el Desarrollo Ágil de Software, los criterios de salida suelen denominarse definición de hecho, y definen las métricas objetivas del equipo para un elemento entregable. Los criterios de entrada que debe cumplir una historia de usuario para iniciar las actividades de desarrollo y/o prueba se denominan Definición de Preparado.

5.1.4 Técnicas de Estimación

La estimación del esfuerzo de prueba consiste en predecir la cantidad de trabajo relacionado con la prueba que se necesita para cumplir los objetivos de un proyecto de prueba. Es importante dejar claro a los implicados que la estimación se basa en una serie de suposiciones y que siempre está sujeta a errores de estimación. La estimación de las tareas pequeñas suele ser más precisa que la de las grandes. Por lo tanto, al estimar una tarea grande, ésta puede descomponerse en un conjunto de tareas más pequeñas que, a su vez, pueden estimarse.

En este programa de estudio se describen las siguientes cuatro técnicas de estimación.

Estimación basada en proporciones.

> En esta técnica basada en métricas, se recopilan cifras de proyectos anteriores de la organización, lo que permite obtener proporciones "estándar" para proyectos similares. Las proporciones de los propios proyectos de una organización (por ejemplo, tomadas de datos históricos) suelen ser la mejor fuente para utilizar en el proceso de estimación. Estas proporciones estándar pueden utilizarse después para estimar el esfuerzo de prueba del nuevo proyecto. Por ejemplo, si en el proyecto anterior la proporción del esfuerzo de desarrollo a prueba era de 3:2, y en el proyecto

actual se espera que el esfuerzo de desarrollo sea de 600 días-persona, el esfuerzo de prueba puede estimarse en 400 días-persona.

Extrapolación.

En esta técnica basada en métricas, las mediciones se realizan lo antes posible en el proyecto actual para recopilar los datos. Al disponer de suficientes observaciones, el esfuerzo necesario para el trabajo restante puede aproximarse extrapolando estos datos (normalmente aplicando un modelo matemático). Este método es muy adecuado en los CVDS iterativos. Por ejemplo, el equipo puede extrapolar el esfuerzo de prueba en la próxima iteración como el esfuerzo medio de las tres últimas iteraciones.

Delphi de Banda Ancha.

En esta técnica iterativa basada en la experiencia, los expertos realizan estimaciones basadas en la experiencia. Cada experto, de forma aislada, estima el esfuerzo. Se recogen los resultados y si hay desviaciones que se salen de los límites acordados, los expertos discuten sus estimaciones actuales. A continuación, se pide a cada experto que haga una nueva estimación basada en esa retroalimentación, de nuevo de forma aislada. Este proceso se repite hasta que se alcanza un consenso. El póker de planificación es una variante del Delphi de Banda Ancha, utilizada habitualmente en el desarrollo ágil de software. En el póker de planificación, las estimaciones suelen hacerse utilizando cartas con números que representan el tamaño del esfuerzo.

Estimación de tres puntos.

En esta técnica basada en expertos, éstos realizan tres estimaciones: la estimación más optimista (a), la estimación más probable (m) y la estimación más pesimista (b). La estimación final (E) es su media aritmética ponderada. En la versión más popular de esta técnica, la estimación se calcula como E = (a + 4*m + b) / 6. La ventaja de esta técnica es que permite a los expertos calcular el error de medición: SD = (b - a) / 6. Por ejemplo, si las estimaciones (en horas-persona) son: a=6, m=9 y b=18, entonces la estimación final es de 10±2 horas-persona (es decir, entre 8 y 12 horas-persona), porque E = (6 + 4*9 + 18) / 6 = 10 y SD = (18 - 6) / 6 = 2.

Véase (Kan 2003, Koomen 2006, Westfall 2009) para estas y muchas otras técnicas de estimación de la prueba.

5.1.5 Priorización de Casos de Prueba

Una vez que los casos de prueba y los procedimientos de prueba se han especificado y reunido en conjuntos de prueba, estos conjuntos de prueba pueden organizarse en un calendario de ejecución de prueba que defina el orden en que deben ejecutarse. A la hora de priorizar los casos de prueba, pueden tenerse en cuenta distintos factores. Las estrategias de priorización de casos de prueba más utilizadas son las siguientes:

- Priorización basada en el riesgo, en la que el orden de ejecución de las pruebas se basa en los resultados del análisis del riesgo (véase la sección 5.2.3). Los casos de prueba que cubren los riesgos más importantes se ejecutan en primer lugar.
- Priorización basada en la cobertura, en la que el orden de ejecución de las pruebas se basa en la cobertura (por ejemplo, la cobertura de sentencias). Los casos de prueba que logran la mayor cobertura se ejecutan en primer lugar. En otra variante, denominada priorización de cobertura adicional, el caso de prueba que logra la cobertura más alta se ejecuta primero; cada caso de prueba posterior es el que logra la cobertura adicional más alta.
- Priorización basada en requisitos, en la que el orden de ejecución de las pruebas se basa en las prioridades de los requisitos trazadas hasta los casos de prueba correspondientes. Las prioridades

de los requisitos son definidas por los implicados. Los casos de prueba relacionados con los requisitos más importantes se ejecutan en primer lugar.

En condiciones ideales, los casos de prueba se ordenarían para su realización en función de sus niveles de prioridad, utilizando, por ejemplo, una de las estrategias de priorización mencionadas. Sin embargo, esta práctica puede no funcionar si los casos de prueba o las prestaciones que se están probando tienen dependencias. Si un caso de prueba con una prioridad más alta depende de un caso de prueba con una prioridad más baja, el caso de prueba de prioridad más baja debe ejecutarse primero.

El orden de ejecución de las pruebas también debe tener en cuenta la disponibilidad de recursos. Por ejemplo, las herramientas de prueba necesarias, los entornos de prueba o las personas que sólo pueden estar disponibles durante un periodo de tiempo específico.

5.1.6 Pirámide de Prueba

La pirámide de prueba es un modelo que muestra que las distintas pruebas pueden tener una granularidad diferente. El modelo de pirámide de prueba apoya al equipo en la automatización de la prueba y en la asignación del esfuerzo de prueba mostrando que los diferentes objetivos se apoyan en diferentes niveles de automatización de la prueba. Las capas de la pirámide representan grupos de pruebas. Cuanto más alta es la capa, menor es la granularidad de la prueba, el aislamiento de la prueba y el tiempo de ejecución de la prueba. Las pruebas de la capa inferior son pequeñas, aisladas, rápidas y comprueban una pequeña parte de la funcionalidad, por lo que normalmente se necesitan muchas para lograr una cobertura razonable. La capa superior representa pruebas complejas, de alto nivel y de extremo a extremo. Estas pruebas de alto nivel suelen ser más lentas que las pruebas de las capas inferiores y suelen comprobar una gran parte de la funcionalidad, por lo que normalmente sólo se necesitan unas pocas para lograr una cobertura razonable. El número y la nomenclatura de las capas pueden diferir. Por ejemplo, el modelo original de pirámide de prueba (Cohn 2009) define tres capas: "pruebas unitarias", "pruebas de servicio" y "pruebas de interfaz de usuario". Otro modelo popular define pruebas unitarias (de componentes), pruebas de integración (integración de componentes) y pruebas de extremo a extremo. También pueden utilizarse otros niveles de prueba (véase la sección 2.2.1).

5.1.7 Cuadrantes de Prueba

Los cuadrantes de prueba, definidos por Brian Marick (Marick 2003, Crispin 2008), agrupan los niveles de prueba con los tipos de prueba, actividades, técnicas de prueba y productos de trabajo apropiados en el Desarrollo Ágil de software. El modelo ayuda a la gestión de pruebas a visualizarlas para asegurar que se incluyen todos los tipos de prueba y niveles de prueba apropiados en el CVDS y a comprender que algunos tipos de prueba son más relevantes para determinados niveles de prueba que otros. Este modelo también proporciona una forma de diferenciar y describir los tipos de pruebas a todos los implicados, incluidos desarrolladores, probadores y representantes de negocio.

En este modelo, las pruebas pueden estar orientadas al negocio o a la tecnología. Las pruebas también pueden apoyar al equipo (es decir, guiar el desarrollo) o criticar el producto (es decir, medir su comportamiento frente a las expectativas). La combinación de estos dos puntos de vista determina los cuatro cuadrantes:

- Cuadrante Q1 (orientado a la tecnología, apoya al equipo). Este cuadrante contiene pruebas de componentes y de integración de componentes. Estas pruebas deberían automatizarse e incluirse en el proceso de IC.
- Cuadrante Q2 (de cara al negocio, apoya al equipo). Este cuadrante contiene pruebas funcionales, ejemplos, pruebas de historias de usuario, prototipos de experiencia de usuario, pruebas de API y simulaciones. Estas pruebas comprueban los criterios de aceptación y pueden ser manuales o automatizadas.

- Cuadrante Q3 (de cara al negocio, critica al producto). Este cuadrante contiene prueba exploratoria, prueba de usabilidad y prueba de aceptación de usuario. Estas pruebas están orientadas al usuario y suelen ser manuales.
- Cuadrante Q4 (de cara a la tecnología, critica al producto). Este cuadrante contiene pruebas de humo y pruebas no funcionales (excepto las pruebas de usabilidad). Estas pruebas suelen estar automatizadas.

5.2 Gestión del Riesgo

Las organizaciones se enfrentan a muchos factores internos y externos que hacen que sea incierto si alcanzarán sus objetivos y cuándo lo harán (ISO 31000). La gestión del riesgo permite a las organizaciones aumentar la probabilidad de alcanzar los objetivos, mejorar la calidad de sus productos y aumentar la confianza de los implicados.

Las principales actividades de gestión del riesgo son:

- Análisis del riesgo (consistente en la identificación del riesgo y la evaluación del riesgo; véase la sección 5.2.3)
- El control del riesgo (que consiste en la mitigación del riesgo y la monitorización del riesgo; véase la sección 5.2.4)

El enfoque de prueba, en el que las actividades de prueba se seleccionan, priorizan y gestionan en función del análisis del riesgo y el control del riesgo, se denomina pruebas basadas en el riesgo.

5.2.1 Definición del Riesgo y Atributos del Riesgo

El riesgo es un acontecimiento, peligro, amenaza o situación potencial cuya ocurrencia causa un efecto adverso. Un riesgo puede caracterizarse por dos factores:

- Probabilidad del riesgo - la probabilidad de que se produzca el riesgo (mayor que cero y menor que uno).
- Impacto del riesgo (daño) - las consecuencias de que se produzca.

Estos dos factores expresan el nivel de riesgo, que es una medida del riesgo. Cuanto mayor sea el nivel de riesgo, más importante será su tratamiento.

5.2.2 Riesgos de Proyecto y Riesgos de Producto

En la prueba de software suelen preocupar dos tipos de riesgos: los riesgos de proyecto y los riesgos de producto.

Los **riesgos de proyecto** están relacionados con la gestión y el control del proyecto. Los riesgos de proyecto incluyen:

- Problemas de organización (por ejemplo, retrasos en las entregas de los productos de trabajo, estimaciones inexactas, reducción de costes).
- Problemas de personal (por ejemplo, competencias insuficientes, conflictos, problemas de comunicación, escasez de personal).
- Problemas técnicos (por ejemplo, corrupción del alcance, soporte deficiente de las herramientas).
- Problemas con los proveedores (por ejemplo, fallo en la entrega por parte de terceros, quiebra de la empresa de apoyo).

Los riesgos de proyecto, cuando se producen, pueden repercutir en el calendario, el presupuesto o el alcance del proyecto, lo que afecta a la capacidad de éste para alcanzar sus objetivos.

Los riesgos de producto están relacionados con las características de calidad del producto (por ejemplo, las descritas en el modelo de calidad ISO 25010). Algunos ejemplos de riesgos de producto son: falta de funcionalidad o funcionalidad incorrecta, cálculos erróneos, errores de ejecución, arquitectura deficiente, algoritmos ineficaces, tiempo de respuesta inadecuado, mala experiencia de usuario, vulnerabilidades de

seguridad. Los riesgos de producto, cuando se producen, pueden dar lugar a diversas consecuencias negativas, entre las que se incluyen:

- Insatisfacción del usuario.
- Pérdida de ingresos, confianza, reputación.
- Daños a terceros.
- Altos costes de mantenimiento, sobrecarga del servicio de asistencia técnica.
- Sanciones penales.
- En casos extremos, daños físicos, lesiones o incluso la muerte.

5.2.3 Análisis del Riesgo de Producto

Desde el punto de vista de la prueba, el objetivo del análisis del riesgo de producto es proporcionar una conciencia del riesgo de producto para concentrar el esfuerzo de prueba de forma que se minimice el nivel residual de riesgo de producto. Lo ideal es que el análisis del riesgo de producto comience en una fase temprana del CVDS.

El análisis del riesgo de producto consiste en la identificación del riesgo y la evaluación del riesgo. La identificación del riesgo consiste en generar una lista exhaustiva de riesgos. Los implicados pueden identificar los riesgos utilizando diversas técnicas y herramientas, por ejemplo, tormenta de ideas, talleres, entrevistas o diagramas causa-efecto. La evaluación del riesgo implica: categorizar los riesgos identificados, determinar su probabilidad de riesgo, su impacto y nivel de riesgo, establecer prioridades y proponer formas de gestionarlos. La categorización ayuda a asignar acciones de mitigación, porque normalmente los riesgos que caen en la misma categoría pueden mitigarse utilizando un enfoque similar.

La evaluación del riesgo puede utilizar un enfoque cuantitativo o cualitativo, o una mezcla de ambos. En el enfoque cuantitativo el nivel de riesgo se calcula como la multiplicación de la probabilidad del riesgo y el impacto del riesgo. En el enfoque cualitativo, el nivel de riesgo puede determinarse utilizando una matriz de riesgo.

El análisis del riesgo de producto puede influir en la minuciosidad y el alcance de las pruebas. Sus resultados se utilizan para:

- Determinar el alcance de la prueba que se debe llevar a cabo.
- Determinar los niveles de prueba concretos y proponer los tipos de prueba que deben realizarse
- Determinar las técnicas de prueba que deben emplearse y la cobertura que debe alcanzarse
- Estimar el esfuerzo de prueba necesario para cada tarea
- Priorizar las pruebas para intentar encontrar los defectos críticos lo antes posible
- Determine si podría emplearse alguna actividad adicional a las pruebas para reducir el riesgo

5.2.4 Control del Riesgo de Producto

El control del riesgo de producto comprende todas las medidas que se toman en respuesta a los riesgos de producto identificados y evaluados. El control del riesgo del producto consiste en la mitigación del riesgo y la monitorización del riesgo. La mitigación del riesgo implica la implementación de las acciones propuestas en la evaluación del riesgo para reducir el nivel de riesgo. El objetivo de la monitorización del riesgo es asegurar que las acciones de mitigación son efectivas, obtener más información para mejorar la evaluación del riesgo e identificar los riesgos emergentes.

Con respecto al control del riesgo de producto, una vez analizado el riesgo, son posibles varias opciones de respuesta al mismo, por ejemplo, mitigación del riesgo mediante la prueba, aceptación del riesgo, transferencia del riesgo o plan de contingencia (Veenendaal 2012). Las medidas que pueden tomarse para mitigar los riesgos de producto mediante la prueba son las siguientes:

- Seleccionar a los probadores con el nivel adecuado de experiencia y competencia, apropiados para un tipo de riesgo determinado.
- Aplicar un nivel adecuado de independencia de la prueba.
- Realizar revisiones y análisis estático.
- Aplicar las técnicas de prueba y los niveles de cobertura adecuados.
- Aplicar los tipos de prueba adecuados que aborden las características de calidad afectadas.
- Realizar pruebas dinámicas, incluidas las pruebas de regresión.

5.3 Monitorización de la Prueba, Control de la Prueba y Compleción de la Prueba

La monitorización de la prueba se ocupa de recopilar información sobre la prueba. Esta información se utiliza para evaluar el avance de la prueba y medir si se satisfacen los criterios de salida de la prueba o las tareas de prueba asociadas a los criterios de salida, como el cumplimiento de los objetivos de cobertura de los riesgos del producto, los requisitos o los criterios de aceptación.

El control de la prueba utiliza la información de la monitorización de prueba para proporcionar, en forma de directivas de control, la orientación y las acciones correctivas necesarias para lograr la prueba más eficaz y eficiente. Algunos ejemplos de directivas de control son:

- Volver a priorizar las pruebas cuando un riesgo identificado se convierte en un problema.
- Reevaluar si un elemento de prueba cumple los criterios de entrada o los criterios de salida debido a la repetición del trabajo.
- Ajustar el calendario de prueba para hacer frente a un retraso en la entrega del entorno de prueba.
- Añadir nuevos recursos cuando y donde se necesiten.

La compleción de la prueba recopila datos de las actividades de prueba finalizadas para consolidar la experiencia, el producto de prueba y cualquier otra información relevante. Las actividades de compleción de la prueba se producen en los hitos del proyecto, como cuando se completa un nivel de prueba, se termina una iteración ágil, se completa (o cancela) un proyecto de prueba, se libera un sistema de software o se completa una entrega de mantenimiento.

5.3.1 Métricas Utilizadas en la Prueba

Las métricas de prueba se recopilan para mostrar el avance con respecto al calendario de prueba y el presupuesto previstos, la calidad actual del objeto de prueba y la efectividad de las actividades de prueba con respecto a los objetivos o a una meta de iteración. La monitorización de prueba recopila una variedad de métricas para apoyar el control de prueba y la compleción de la prueba.

Entre las métricas de prueba más comunes se incluyen:

- Métricas de avance del proyecto (por ejemplo, compleción de la tarea, uso de recursos, esfuerzo de la prueba).
- Métricas de avance de la prueba (por ejemplo, avance en la implementación de casos de prueba, avance en la preparación del entorno de prueba, número de casos de prueba realizados/no realizados, pasados/fallados, tiempo de ejecución de prueba).
- Métricas de calidad de producto (por ejemplo, disponibilidad, tiempo de respuesta, tiempo medio hasta el fallo).
- Métricas de defectos (por ejemplo, número y prioridades de defectos encontrados/corregidos, densidad de defectos, porcentaje de detección de defectos).
- Métricas de riesgo (por ejemplo, nivel de riesgo residual).
- Métricas de cobertura (por ejemplo, cobertura de requisitos, cobertura de código).
- Métricas de coste (por ejemplo, coste de la prueba, coste organizativo de la calidad).

5.3.2 Propósito, Contenido y Audiencia de los Informes de Prueba

El suministro de información de prueba resume y comunica la información de prueba durante y después de la prueba. Los informes del avance de la prueba apoyan el control continuo de la prueba y deben proporcionar suficiente información para realizar modificaciones en el calendario de prueba, los recursos o el plan de prueba, cuando dichos cambios sean necesarios debido a una desviación del plan o a un cambio de circunstancias. Los informes de compleción de la prueba resumen una etapa específica de la prueba (por ejemplo, nivel de prueba, ciclo de prueba, iteración) y pueden dar información para pruebas posteriores.

Durante la monitorización y el control de prueba, el equipo de prueba genera informes del avance de la prueba para los implicados con el fin de mantenerlos informados. Los informes del avance de la prueba suelen generarse de forma regular (por ejemplo, a diario, semanalmente, etc.) e incluyen:

- Periodo de prueba.
- Avance de la prueba (por ejemplo, por delante o por detrás de lo previsto), incluyendo cualquier desviación notable.
- Obstáculos para la prueba y sus soluciones.
- Métricas de prueba (véanse ejemplos en la sección 5.3.1).
- Riesgos nuevos y modificados dentro del periodo de prueba.
- Prueba planificada para el siguiente periodo.

Un informe compleción de prueba se prepara durante la finalización de la prueba, cuando un proyecto, nivel de prueba o tipo de prueba está completo y cuando, idealmente, se han cumplido sus criterios de salida. Este informe utiliza los informes del avance de la prueba y otros datos. Los informes típicos de compleción de la prueba incluyen:

- Resumen de la prueba.
- Evaluación de la prueba y de la calidad del producto basada en el plan de prueba original (es decir, objetivos de la prueba y criterios de salida).
- Desviaciones del plan de prueba (por ejemplo, diferencias con el calendario, la duración y el esfuerzo previstos).
- Obstáculos y soluciones a las pruebas.
- Métricas de prueba basadas en los informes del avance de la prueba.
- Riesgos no mitigados, defectos no solucionados.
- Lecciones aprendidas relevantes para la prueba.

Distintos públicos requieren información diferente en los informes e influyen en el grado de formalidad y la frecuencia de los mismos. Informar sobre el avance de la prueba a otras personas del mismo equipo suele ser frecuente e informal, mientras que informar sobre la prueba de un proyecto finalizado sigue una plantilla establecida y sólo se produce una vez.

El estándar ISO/IEC/IEEE 29119-3 incluye plantillas y ejemplos de informes del avance de la prueba (denominados informes del estado de la prueba) e informes de compleción de la prueba.

5.3.3 Comunicación del Estado de la Prueba

La mejor forma de comunicar el estado de la prueba varía en función de los asuntos de interés de la gestión de la prueba, las estrategias de prueba de la organización, las normas reglamentarias o, en el caso de los equipos autoorganizados (véase la sección 1.5.2), del propio equipo. Las opciones incluyen:

- Comunicación verbal con los miembros del equipo y otros implicados.
- Cuadros de mando (por ejemplo, paneles de control de CI/CD, tableros de tareas y gráficos de quemado).
- Canales de comunicación electrónica (por ejemplo, correo electrónico, chat).
- Documentación en línea.
- Informes formales de prueba (véase la sección 5.3.2).

Pueden utilizarse una o varias de estas opciones. Una comunicación más formal puede ser más apropiada para equipos distribuidos en los que la comunicación directa cara a cara no siempre es posible debido a la distancia geográfica o a las diferencias horarias. Normalmente, los distintos implicados están interesados en diferentes tipos de información, por lo que la comunicación debe adaptarse en consecuencia.

5.4 Gestión de la Configuración

En las pruebas, la gestión de la configuración (GC) proporciona una disciplina para la identificación, el control y el seguimiento de productos de trabajo como planes de prueba, estrategias de prueba, condiciones de prueba, casos de prueba, guiones de prueba, resultados de prueba, registros de prueba e informes de prueba como elementos de la configuración.

Para un elemento de la configuración complejo (por ejemplo, un entorno de prueba), la GC registra los elementos que lo componen, sus relaciones y versiones. Si el elemento de la configuración se aprueba para ser probado, se convierte en una línea base y sólo puede modificarse mediante un proceso formal de control de cambios.

La gestión de la configuración mantiene un registro de los elementos de la configuración modificados cuando se crea una nueva línea base. Es posible volver a una línea base anterior para reproducir los resultados de pruebas anteriores.

Para apoyar adecuadamente las pruebas, la gestión de la configuración asegura lo siguiente:

- Todos los elementos de configuración, incluidos los elementos de prueba (partes individuales del objeto de prueba), se identifican de forma única, se controla su versión, se realiza un seguimiento de los cambios y se relacionan con otros elementos de configuración para que pueda mantenerse la trazabilidad durante todo el proceso de prueba.
- Todos los elementos de documentación y software identificados se referencian de forma inequívoca en la documentación de prueba.

La integración continua, la entrega continua, el despliegue continuo y las pruebas asociadas suelen implementarse como parte de una canalización automatizada de DevOps (véase la sección 2.1.4), en la que normalmente se incluye la GC automatizada.

5.5 Gestión de Defectos

Dado que uno de los principales objetivos de prueba es encontrar defectos, es esencial contar con un proceso establecido de gestión de defectos. Aunque aquí nos referimos a "defectos", las anomalías notificadas pueden resultar ser defectos reales u otra cosa (por ejemplo, un falso positivo, una solicitud de cambio); esto se resuelve durante el proceso de tratamiento de los informes de defecto. Las anomalías pueden informarse durante cualquier fase del CVDS y la forma depende de éste. Como mínimo, el proceso de gestión de defectos incluye un flujo de trabajo para tratar las anomalías individuales desde su descubrimiento hasta su cierre y reglas para su clasificación. El flujo de trabajo suele incluir actividades para registrar las anomalías notificadas, analizarlas y clasificarlas, decidir una respuesta adecuada como arreglarlas o mantenerlas tal cual y, por último, cerrar el informe de defecto. El proceso debe ser seguido por todos los implicados. Es aconsejable tratar los defectos de las pruebas estáticas (especialmente el análisis estático) de forma similar.

Los informes de defectos típicos tienen los siguientes objetivos:

- Proporcionar a los responsables de la gestión y resolución de los defectos notificados información suficiente para resolver el problema.
- Proporcionar un medio de seguimiento de la calidad del producto del trabajo.
- Proporcionar ideas para la mejora del proceso de desarrollo y prueba.

Un informe de defecto registrado durante una prueba dinámica suele incluir:

- Identificador único.
- Título con un breve resumen de la anomalía que se informa.
- Fecha en que se observó la anomalía, organización emisora y autor, incluido su rol.
- Identificación del objeto de prueba y del entorno de prueba.
- Contexto del defecto (por ejemplo, caso de prueba que se está realizando, actividad de prueba que se está llevando a cabo, fase del CVDS y otra información relevante como la técnica de prueba, la lista de comprobación o los datos de prueba que se están utilizando).
- Descripción del fallo para permitir su reproducción y resolución, incluidos los pasos que detectaron la anomalía, y cualquier registro de prueba, volcado de la base de datos, captura de pantalla o grabación pertinentes.
- Resultados esperados y resultados reales.
- Severidad del defecto (grado de impacto) sobre los intereses de los implicados o los requisitos.
- Prioridad de corrección.
- Estado del defecto (por ejemplo, abierto, aplazado, duplicado, a la espera de ser corregido, a la espera de pruebas de confirmación, reabierto, cerrado, rechazado).
- Referencias (por ejemplo, al caso de prueba).

Algunos de estos datos pueden incluirse automáticamente al utilizar herramientas de gestión de defectos (por ejemplo, identificador, fecha, autor y estado inicial). En el estándar ISO/IEC/IEEE 29119-3, que se refiere a los informes de defectos como informes de incidencias, se pueden encontrar plantillas de documentos para un informe de defectos y ejemplos de informes de defectos.

6 Herramientas de Prueba - 20 minutos

Duración: 20 minutos

Palabras Clave

Español	Inglés
automatización de la prueba	test automation

Objetivos de Aprendizaje para "Capítulo 6"

6.1 Herramientas de Apoyo a la Prueba

FL-6.1.1	(K2)	Explicar cómo los distintos tipos de herramientas de prueba dan soporte a la prueba

6.2 Ventajas y Riesgos de la Automatización de la Prueba

FL-6.2.1	(K1)	Recordar las ventajas y los riesgos de la automatización de la prueba

6.1 Herramientas de Apoyo a la Prueba

Las herramientas de prueba apoyan y facilitan muchas actividades de prueba. Algunos ejemplos son

- Herramientas de gestión - aumentan la eficiencia del proceso de prueba facilitando la gestión del CVDS, los requisitos, las pruebas, los defectos y la configuración.
- Herramientas de prueba estática: ayudan al probador a realizar revisiones y análisis estáticos.
- Herramientas de diseño de pruebas e implementación: facilitan la generación de casos de prueba, datos de prueba y procedimientos de prueba.
- Herramientas de ejecución de la prueba y de cobertura: facilitan la ejecución automatizada de la prueba y la medición de la cobertura.
- Herramientas de pruebas no funcionales - permiten al probador realizar pruebas no funcionales que son difíciles o imposibles de realizar manualmente
- Herramientas DevOps - apoyan la canalización de entrega DevOps, el seguimiento del flujo de trabajo, los procesos de construcción automatizados, IC/EC
- Herramientas de colaboración - facilitan la comunicación
- Herramientas de apoyo a la escalabilidad y la normalización del despliegue (por ejemplo, máquinas virtuales, herramientas de contenerización[8])
- Cualquier otra herramienta que ayude en las pruebas (por ejemplo, una hoja de cálculo es una herramienta de prueba en el contexto de las pruebas)

[8] "contenerización" es la traducción del término "containerization". Esta traducción no es definitiva.

6.2 Ventajas y Riesgos de la Automatización de la Prueba

La mera adquisición de una herramienta no garantiza el éxito. Cada nueva herramienta requerirá un esfuerzo para lograr beneficios reales y duraderos (por ejemplo, para la introducción de la herramienta, el mantenimiento y la formación). También existen algunos riesgos, que necesitan análisis y mitigación.

Entre los beneficios potenciales de utilizar la automatización de la prueba se incluyen:

- Ahorro de tiempo al reducir el trabajo manual repetitivo (por ejemplo, ejecutar pruebas de regresión, volver a introducir los mismos datos de prueba, comparar los resultados esperados frente a los resultados reales y cotejarlos con los estándares de codificación).
- Prevención de errores humanos simples gracias a una mayor consistencia y repetibilidad (por ejemplo, las pruebas se obtienen de forma coherente a partir de los requisitos, los datos de prueba se crean de forma sistemática y las pruebas se ejecutan mediante una herramienta en el mismo orden y con la misma frecuencia).
- Evaluación más objetiva (por ejemplo, la cobertura) y proporcionar medidas que son demasiado complicadas de obtener para los humanos
- Acceso más fácil a la información sobre pruebas para apoyar la gestión de pruebas y la elaboración de informes de pruebas (por ejemplo, estadísticas, gráficos y datos agregados sobre el avance de las pruebas, las tasas de defectos y la duración de la ejecución de las pruebas).
- Tiempos de ejecución de pruebas reducidos para proporcionar una detección de defectos más temprana, una retroalimentación más rápida y un tiempo de comercialización más rápido
- Más tiempo para que los probadores diseñen pruebas nuevas, más profundas y más eficaces

Entre los riesgos potenciales de utilizar la automatización de la prueba se incluyen:

- Expectativas poco realistas sobre las ventajas de una herramienta (incluidas la funcionalidad y la facilidad de uso).
- Estimaciones imprecisas del tiempo, los costes y el esfuerzo necesarios para introducir una herramienta, mantener los guiones de prueba y cambiar el proceso de prueba manual existente.
- Utilizar una herramienta de prueba cuando la prueba manual es más apropiada.
- Confiar demasiado en una herramienta, por ejemplo, ignorando la necesidad del pensamiento crítico humano.
- La dependencia del proveedor de la herramienta, que puede quebrar, retirar la herramienta, venderla a otro proveedor o proporcionar un soporte deficiente (por ejemplo, respuestas a consultas, actualizaciones y corrección de defectos).
- Utilizar un software de código abierto que puede estar abandonado, lo que significa que no hay más actualizaciones disponibles, o sus componentes internos pueden requerir actualizaciones bastante frecuentes como desarrollo posterior.
- La herramienta de automatización no es compatible con la plataforma de desarrollo.
- Elección de una herramienta inadecuada que no cumpla los requisitos reglamentarios y/o los estándares de seguridad física.

7 Referencias

6.3 Estándares

ISO/IEC/IEEE 29119-1 (2022) Software and systems engineering – Software testing – Part 1: General Concepts

ISO/IEC/IEEE 29119-2 (2021) Software and systems engineering – Software testing – Part 2: Test processes

ISO/IEC/IEEE 29119-3 (2021) Software and systems engineering – Software testing – Part 3: Test documentation

ISO/IEC/IEEE 29119-4 (2021) 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

ISO/IEC/IEEE 14764:2022 – Software engineering – Software life cycle processes – Maintenance ISO 31000 (2018) Risk management – Principles and guidelines

6.4 Libros

Adzic, G. (2009) Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing, Neuri Limited

Ammann, P. and Offutt, J. (2016) Introduction to Software Testing (2e), Cambridge University Press

Andrews, M. and Whittaker, J. (2006) How to Break Web Software: Functional and Security Testing of Web Applications and Web Services, Addison-Wesley Professional

Beck, K. (2003) Test Driven Development: By Example, Addison-Wesley

Beizer, B. (1990) Software Testing Techniques (2e), Van Nostrand Reinhold: Boston MA Boehm, B. (1981) Software Engineering Economics, Prentice Hall, Englewood Cliffs, NJ

Buxton, J.N. and Randell B., eds (1970), Software Engineering Techniques. Report on a conference sponsored by the NATO Science Committee, Rome, Italy, 27–31 October 1969, p. 16

Chelimsky, D. et al. (2010) The Rspec Book: Behaviour Driven Development with Rspec, Cucumber, and Friends, The Pragmatic Bookshelf: Raleigh, NC

Cohn, M. (2009) Succeeding with Agile: Software Development Using Scrum, Addison-Wesley 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: A Practical Guide for Testers and Agile Teams, Pearson Education: Boston MA

Forgács, I., and Kovács, A. (2019) Practical Test Design: Selection of traditional and automated test design techniques, BCS, The Chartered Institute for IT

Gawande A. (2009) The Checklist Manifesto: How to Get Things Right, New York, NY: Metropolitan Books

Gärtner, M. (2011), ATDD by Example: A Practical Guide to Acceptance Test-Driven Development, Pearson Education: Boston MA

Gilb, T., Graham, D. (1993) Software Inspection, Addison Wesley

Hendrickson, E. (2013) Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing, The Pragmatic Programmers

Hetzel, B. (1988) The Complete Guide to Software Testing, 2nd ed., John Wiley and Sons

Jeffries, R., Anderson, A., Hendrickson, C. (2000) Extreme Programming Installed, Addison-Wesley Professional

Jorgensen, P. (2014) Software Testing, A Craftsman's Approach (4e), CRC Press: Boca Raton FL Kan, S. (2003) Metrics and Models in Software Quality Engineering, 2nd ed., Addison-Wesley Kaner, C., Falk, J., and Nguyen, H.Q. (1999) Testing Computer Software, 2nd ed., Wiley

Kaner, C., Bach, J., and Pettichord, B. (2011) Lessons Learned in Software Testing: A Context-Driven Approach, 1st ed., Wiley

Kim, G., Humble, J., Debois, P. and Willis, J. (2016) The DevOps Handbook, Portland, OR

Koomen, T., van der Aalst, L., Broekman, B. and Vroon, M. (2006) TMap Next for result-driven testing, UTN Publishers, The Netherlands

Myers, G. (2011) The Art of Software Testing, (3e), John Wiley & Sons: New York NY O'Regan, G. (2019) Concise Guide to Software Testing, Springer Nature Switzerland Pressman, R.S. (2019) Software Engineering. A Practitioner's Approach, 9th ed., McGraw Hill

Roman, A. (2018) Thinking-Driven Testing. The Most Reasonable Approach to Quality Control, Springer Nature Switzerland

Van Veenendaal, E (ed.) (2012) Practical Risk-Based Testing, The PRISMA Approach, UTN Publishers: The Netherlands

Watson, A.H., Wallace, D.R. and McCabe, T.J. (1996) Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, U.S. Dept. of Commerce, Technology Administration, NIST

Westfall, L. (2009) The Certified Software Quality Engineer Handbook, ASQ Quality Press Whittaker, J. (2002) How to Break Software: A Practical Guide to Testing, Pearson

Whittaker, J. (2009) Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design, Addison Wesley

Whittaker, J. and Thompson, H. (2003) How to Break Software Security, Addison Wesley Wiegers, K. (2001) Peer Reviews in Software: A Practical Guide, Addison-Wesley Professional

Gawande A. (2009) The Checklist Manifesto: How to Get Things Right, New York, NY: Metropolitan Books

Gärtner, M. (2011), ATDD by Example: A Practical Guide to Acceptance Test-Driven Development, Pearson Education: Boston MA

Gilb, T., Graham, D. (1993) Software Inspection, Addison Wesley

Hendrickson, E. (2013) Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing, The Pragmatic Programmers

Hetzel, B. (1988) The Complete Guide to Software Testing, 2nd ed., John Wiley and Sons

Jeffries, R., Anderson, A., Hendrickson, C. (2000) Extreme Programming Installed, Addison-Wesley Professional

Jorgensen, P. (2014) Software Testing, A Craftsman's Approach (4e), CRC Press: Boca Raton FL Kan, S. (2003) Metrics and Models in Software Quality Engineering, 2nd ed., Addison-Wesley Kaner, C., Falk, J., and Nguyen, H.Q. (1999) Testing Computer Software, 2nd ed., Wiley

Kaner, C., Bach, J., and Pettichord, B. (2011) Lessons Learned in Software Testing: A Context-Driven Approach, 1st ed., Wiley

Kim, G., Humble, J., Debois, P. and Willis, J. (2016) The DevOps Handbook, Portland, OR

Koomen, T., van der Aalst, L., Broekman, B. and Vroon, M. (2006) TMap Next for result-driven testing, UTN Publishers, The Netherlands

Myers, G. (2011) The Art of Software Testing, (3e), John Wiley & Sons: New York NY O'Regan, G. (2019) Concise Guide to Software Testing, Springer Nature Switzerland Pressman, R.S. (2019) Software Engineering. A Practitioner's Approach, 9th ed., McGraw Hill

Roman, A. (2018) Thinking-Driven Testing. The Most Reasonable Approach to Quality Control, Springer Nature Switzerland

Van Veenendaal, E (ed.) (2012) Practical Risk-Based Testing, The PRISMA Approach, UTN Publishers: The Netherlands

Watson, A.H., Wallace, D.R. and McCabe, T.J. (1996) Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, U.S. Dept. of Commerce, Technology Administration, NIST

Westfall, L. (2009) The Certified Software Quality Engineer Handbook, ASQ Quality Press Whittaker, J. (2002) How to Break Software: A Practical Guide to Testing, Pearson

Whittaker, J. (2009) Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design, Addison Wesley

Whittaker, J. and Thompson, H. (2003) How to Break Software Security, Addison Wesley Wiegers, K. (2001) Peer Reviews in Software: A Practical Guide, Addison-Wesley Professional

Wiegers, K. (2001) Peer Reviews in Software: A Practical Guide, Addison-Wesley Professional

6.5 Artículos y Páginas Web

Brykczynski, B. (1999) "A survey of software inspection checklists," *ACM SIGSOFT Software Engineering Notes, 24(1), pp. 82-89*

Enders, A. (1975) "An Analysis of Errors and Their Causes in System Programs," *IEEE Transactions on Software Engineering 1(2), pp. 140-149*

Manna, Z., Waldinger, R. (1978) "The logic of computer programming," *IEEE Transactions on Software Engineering 4(3), pp. 199-229*

Marick, B. (2003) Exploration through Example, http://www.exampler.com/old- blog/2003/08/21.1.html#agile-testing-project-1

Nielsen, J. (1994) "Enhancing the explanatory power of usability heuristics," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating Interdependence, ACM Press, pp. 152–158*

Salman. I. (2016) "Cognitive biases in software quality and testing," *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16), ACM, pp. 823-826.*

Wake, B. (2003) "INVEST in Good Stories, and SMART Tasks," https://xp123.com/articles/invest-in-good- stories-and-smart-tasks/

6.6 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

8 Apéndice A - Objetivos de Aprendizaje/Nivel Cognitivo de Conocimiento

Los siguientes objetivos de aprendizaje se definen como aplicables a este programa de estudio. Cada tema del programa de estudio se examinará de acuerdo con el objetivo de aprendizaje correspondiente. Los objetivos de aprendizaje comienzan con un verbo de acción correspondiente a su nivel cognitivo de conocimiento, como se indica a continuación.

Nivel 1: Recordar (K1): el candidato recordará, reconocerá y rememorará un término o concepto.

Verbos de acción: identificar, recordar, rememorar, reconocer.

Ejemplos:

- "Identificar los objetivos característicos de la prueba".
- "Recordar los conceptos de la pirámide de prueba".
- "Reconocer cómo un probador añade valor a la planificación de la iteración y la entrega".

Nivel 2: Comprender (K2): el candidato puede seleccionar las razones o explicaciones de los enunciados relacionados con el tema y puede resumir, comparar, clasificar y dar ejemplos para el concepto de prueba.

Verbos de acción: clasificar, comparar, contrastar, diferenciar, distinguir, ejemplificar, explicar, dar ejemplos, interpretar, resumir.

Ejemplos:

- " Clasificar las diferentes opciones para redactar los criterios de aceptación".
- " Comparar los diferentes roles en la prueba" (buscar similitudes, diferencias o ambas).
- " Distinguir entre riesgos de proyecto y riesgos de producto" (permite diferenciar conceptos).
- "Dar ejemplos de la finalidad y el contenido de un plan de prueba".
- " Explicar el impacto del contexto en el proceso de prueba".
- " Resumir las actividades del proceso de revisión".

Nivel 3: Aplicar (K3): el candidato puede llevar a cabo un procedimiento cuando se enfrenta a una tarea conocida, o seleccionar el procedimiento correcto y aplicarlo a un contexto determinado.

Verbos de acción: aplicar, implementar, preparar, utilizar.

Ejemplos:

- "Aplicar la priorización de casos de prueba" (debe referirse a un procedimiento, técnica, proceso, algoritmo, etc.).
- " Preparar un informe de defectos".
- " Utilizar el análisis del valor frontera para obtener casos de prueba".

Referencias para los niveles cognitivos de los objetivos de aprendizaje

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

9 Apéndice B - Matriz de Trazabilidad de los Resultados de Negocio con respecto a los Objetivos de Aprendizaje

Esta sección enumera el número de objetivos de aprendizaje de nivel básico relacionados con los resultados de negocio y la trazabilidad entre los resultados de negocio de nivel básico y los objetivos de aprendizaje de nivel básico.

Resultados de Negocio: Nivel Básico		FL - BO - 01	FL - BO - 02	FL - BO - 03	FL - BO - 04	FL - BO - 05	FL - BO - 06	FL - BO - 07	FL - BO - 08	FL - BO - 09	FL - BO - 10	FL - BO - 11	FL - BO - 12	FL - BO - 13	FL - BO - 14
BO - 01	Comprender qué es probar y por qué es beneficioso	6													
BO - 02	Comprender los conceptos fundamentales de la prueba de software		22												
BO - 03	Identificar el enfoque de prueba y las actividades que se deben implementar en función del contexto de prueba			6											
BO - 04	Evaluar y mejorar la calidad de la documentación				9										
BO - 05	Aumentar la efectividad y eficiencia de la prueba					20									
BO - 06	Alinear el proceso de prueba con el ciclo de vida de desarrollo de software						6								
BO - 07	Comprender los principios de gestión de la prueba							6							
BO - 08	Redactar y comunicar informes de defectos claros y comprensibles								1						
BO - 09	Comprender los factores que influyen en las prioridades y esfuerzos relacionados con la prueba									7					
BO - 10	Trabajar como parte de un equipo interfuncional										8				
BO - 11	Conocer los riesgos y beneficios relacionados con la automatización de la prueba											1			
BO - 12	Identificar las competencias esenciales necesarias para probar												5		
BO - 13	Comprender el impacto del riesgo en las pruebas													4	
BO - 14	Informar eficazmente sobre el avance y la calidad de la prueba														4

Capítulo/ sección/ subsección	Objetivos de Aprendizaje	Nivel - K	BUSINESS OUTCOMES														
			FL - BO - 01	FL - BO - 02	FL - BO - 03	FL - BO - 04	FL - BO - 05	FL - BO - 06	FL - BO - 07	FL - BO - 08	FL - BO - 09	FL - BO - 10	FL - BO - 11	FL - BO - 12	FL - BO - 13	FL - BO - 14
Capítulo 1	**Fundamentos del Proceso de Prueba**															
1.1	**¿Qué es Probar?**															
1.1.1	Identificar objetivos de prueba característicos.	K1	X													
1.1.2	Diferenciar entre probar y depurar.	K2		X												
1.2	**¿Por qué es Necesario Probar?**															
1.2.1	Aportar ejemplos de por qué es necesario realizar pruebas.	K2	X													
1.2.2	Recordar la relación entre prueba y aseguramiento de la calidad.	K1		X												
1.2.3	Distinguir entre causa raíz, error, defecto y fallo.	K2		X												
1.3	**Principios de la Prueba**															
1.3.1	Explicar los siete principios de la prueba.	K2		X												
1.4	**Actividades de Prueba, Productos de Prueba y Roles de Prueba**															
1.4.1	Resumir las diferentes actividades y tareas de prueba.	K2			X											
1.4.2	Explicar el impacto del contexto en el proceso de prueba.	K2			X			X								
1.4.3	Diferenciar el producto de prueba que soporta las actividades de prueba.	K2			X											
1.4.4	Explicar el valor de mantener la trazabilidad.	K2				X	X									
1.4.5	Comparar los diferentes roles en la prueba.	K2										X				

1.5	**Competencias Esenciales y Buenas Prácticas en la Prueba**															
1.5.1	Dar ejemplos de las competencias genéricas necesarias para probar.	K2											X			
1.5.2	Recordar las ventajas del enfoque de equipo completo.	K1										X				
1.5.3	Distinguir las ventajas e inconvenientes de la independencia de la prueba.	K2			X											
Capítulo 2	**Prueba a lo Largo del Ciclo de Vida de Desarrollo de Software**															
2.1	**La Prueba en el Contexto de un Ciclo de Vida de Desarrollo de Software**															
2.1.1	Explicar el impacto del ciclo de vida de desarrollo de software elegido en la prueba.	K2					X									
2.1.2	Recordar las buenas prácticas de prueba que se aplican a todos los ciclos de vida de desarrollo del software.	K1					X									
2.1.3	Recordar los ejemplos de enfoques de prueba primero para el desarrollo.	K1				X										
2.1.4	Resumir cómo DevOps puede tener un impacto en la prueba.	K2				X	X			X	X					
2.1.5	Explicar el enfoque de desplazamiento a la izquierda.	K2				X	X									
2.1.6	Explicar cómo se pueden utilizar las retrospectivas como mecanismo para la mejora del proceso.	K2				X					X					
2.2	**Niveles de Prueba y Tipos de Prueba**															
2.2.1	Distinguir los diferentes niveles de prueba.	K2		X	X											
2.2.2	Distinguir los diferentes tipos de prueba.	K2		X												
2.2.3	Distinguir la prueba de confirmación de la prueba de regresión.	K2		X												
2.3	**Prueba de Mantenimiento**															
2.3.1	Resumir la prueba de mantenimiento y sus desencadenantes.	K2		X				X								
Capítulo 3	**Prueba Estática**															
3.1	**Prueba estática - Fundamentos**															

3.1.1	Reconocer los tipos de productos que pueden ser evaluados mediante las diferentes técnicas de prueba estática.	K1				X	X										
3.1.2	Explicar el valor de la prueba estática.	K2	X			X	X										
3.1.3	Comparar y contrastar la prueba estática y la prueba dinámica.	K2				X	X										
3.2	**Retroalimentación y Proceso de Revisión**																
3.2.1	Identificar las ventajas de una retroalimentación temprana y frecuente de los implicados.	K1	X			X						X					
3.2.2	Resumir las actividades del proceso de revisión.	K2			X	X											
3.2.3	Recordar qué responsabilidades se asignan a los principales roles a la hora de realizar revisiones.	K1				X								X			
3.2.4	Comparar y contrastar los diferentes tipos de revisión.	K2		X													
3.2.5	Recordar los factores que contribuyen al éxito de una revisión.	K1					X							X			
Capítulo 4	**Análisis y Diseño de la Prueba**																
4.1	**Introducción a las Técnicas de Prueba**																
4.1.1	Distinguir entre técnicas de prueba de caja negra, de caja blanca y basadas en la experiencia.	K2		X													
4.2	**Técnicas de Prueba de Caja Negra**																
4.2.1	Utilizar partición de equivalencia para obtener casos de prueba.	K3					X										
4.2.2	Utilizar análisis del valor frontera para obtener casos de prueba.	K3					X										
4.2.3	Utilizar prueba de tabla de decisión para obtener casos de prueba.	K3					X										
4.2.4	Utilizar prueba de transición de estado para obtener casos de prueba.	K3					X										
4.3	**Técnicas de Prueba de Caja Blanca**																
4.3.1	Explicar la prueba de sentencia.	K2		X													

4.3.2	Explicar la prueba de rama.	K2		X													
4.3.3	Explicar el valor de la prueba de caja blanca.	K2	X	X													
4.4	**Técnicas de Prueba Basadas en la Experiencia**																
4.4.1	Explicar la predicción de errores.	K2		X													
4.4.2	Explicar la prueba exploratoria.	K2		X													
4.4.3	Explicar la prueba basada en lista de comprobación.	K2		X													
4.5	**Enfoques de Prueba Basados en la Colaboración**																
4.5.1	Explicar cómo escribir historias de usuario en colaboración con desarrolladores y representantes de negocio.	K2				X						X					
4.5.2	Clasificar las diferentes opciones para escribir criterios de aceptación.	K2										X					
4.5.3	Utilizar el desarrollo guiado por prueba de aceptación (DGPA) para obtener casos de prueba.	K3					X										
Capítulo 5	**Gestión de la Prueba**																
5.1	**Planificación de la Prueba**																
5.1.1	Dar ejemplos del propósito y el contenido de un plan de prueba.	K2		X					X								
5.1.2	Reconocer cómo un probador añade valor a la planificación de la iteración y de la entrega.	K1	X									X		X			
5.1.3	Comparar y contrastar los criterios de entrada y los criterios de salida.	K2				X		X									X
5.1.4	Utilizar técnicas de estimación para calcular el esfuerzo de prueba necesario.	K3							X		X						
5.1.5	Aplicar la priorización de casos de prueba.	K3							X		X						
5.1.6	Recordar los conceptos de la pirámide de prueba.	K1		X													
5.1.7	Resumir los cuadrantes de prueba y sus relaciones con los niveles de prueba y los tipos de prueba.	K2		X							X						
5.2	**Gestión del Riesgo**																

5.2.1	Identificar el nivel de riesgo utilizando la probabilidad del riesgo y el impacto del riesgo.	K1							X						X	
5.2.2	Distinguir entre riesgos de proyecto y riesgos de producto.	K2		X											X	
5.2.3	Explicar cómo el análisis del riesgo de producto puede influir en la minuciosidad y el alcance de las pruebas.	K2					X				X				X	
5.2.4	Explique qué medidas pueden tomarse en respuesta a los riesgos de producto analizados.	K2		X			X								X	
5.3	**Monitorización de la Prueba, Control de la Prueba y Compleción de la Prueba**															
5.3.1	Recordar las métricas utilizadas para probar.	K1									X					X
5.3.2	Resumir los propósitos, el contenido y las audiencias de los informes de prueba.	K2					X				X					X
5.3.3	Dar ejemplos de cómo comunicar el estado de la prueba.	K2												X		X
5.4	**Gestión de la Configuración**															
5.4.1	Resumir cómo la gestión de la configuración apoya la prueba.	K2					X		X							
5.5	**Gestión de Defectos**															
5.5.1	Preparar un informe de defecto.	K3		X						X						
Capítulo 6	**Herramientas de Prueba**															
6.1	**Herramientas de Apoyo a la Prueba**															
6.1.1	Explicar cómo los distintos tipos de herramientas de prueba dan soporte a la prueba.	K2					X									
6.2	**Ventajas y Riesgos de la Automatización de la Prueba**															
6.2.1	Recordar las ventajas y los riesgos de la automatización de la prueba.	K1					X						X			