Probador Certificado de ISTQB®

Programa de Estudio

Nivel Avanzado

Analista de Prueba

Versión ES V01.00

Traducción realizada por

Spanish Software Testing Qualifications Board

Traducción del Programa de Estudio

"ISTQB® Certified Tester - Advanced Level - Test Analyst (CTAL-TA) - Version V4.0"

SSTQB
Spanish Software Testing Qualifications Board

ISTQB®
International Software Testing Qualifications Board

Nota sobre derechos de propiedad intelectual

Historial de revisiones

Versión	Fecha	Observaciones
v4.0	02/05/2025	Actualización significativa con revisión general y actualización del alcance.
v3.1.2	31/01/2022	Erratas: Corrección de defectos menores de formato, gramática y redacción.
v3.1.1	15/05/2021	Fe de erratas: El aviso de copyright se ha adaptado a los estándares actuales de ISTQB.
v3.1	03/03/2021	Actualización menor con la reescritura de la sección 3.2.3 y varias mejoras de redacción
v3.0 (2019)	19/10/2019	Actualización significativa con revisión general y reducción del alcance.
v2.0 (2012)	19/10/2012	Primera versión como programa de estudio CTAL-TA independiente.

Historial de revisiones – Traducción al idioma español		
Versión	**Fecha**	**Observaciones**
1.00	21/05/2025	No hay observaciones.

Tabla de Contenidos

Agradecimientos

La Asamblea General del ISTQB® publicó formalmente este documento el 1 de mayo de 2025.

Ha sido elaborado por un equipo de “International Software Testing Qualifications Board”: Armin Born, Filipe Carlos, Wim Decoutere, István Forgács, Matthias Hamburg (Product Owner), Attila Kovács, Sandy Liu, François Martin, Stuart Reid, Adam Roman, Jan Sabak, Murian Song, Tanja Tremmel, Marc-Florian Wendland, Tao Xian Feng.

El equipo da las gracias a Julia Sabatine por su corrección de texto, a Gary Mogyorodi por su revisión técnica, a Daniel Pol'an por su constante apoyo técnico, y al equipo revisor y a los Comités Miembro por sus sugerencias y aportaciones.

Las siguientes personas participaron en la revisión, con comentarios y en la votación de este programa de estudio:

Edición actual (v4.0): Gergely Ágnecz, Laura Albert, Dani Almog, Somying Atiporntham, Andre Baumann, Lars Bjørstrup, Ralf Bongard, Earl Burba, Filipe Carlos, Renzo Cerquozzi, Kanitta Chantaramanon, Alessandro Collino, Nicola De Rosa, Aneta Derkova, Dingguofu, Ole Chr. Hansen, Zheng Dandan, Karol Frühauf, Dinu Gamage, Chen Geng, Sabine Gschwandtner, Ole Chr. Hansen, Zsolt Hargitai, Tharushi Hettiarachchi, Ágota Horváth, Arnika Hryszko, Caroline Julien, Beata Karpinska, Ramit Manohar Kaul, Mattijs Kemmink, László Kvintovics, Thomas Letzkus, Marek Majerník, Donald Marcotte, Dénes Medzihradszky, Imre Mészáros, Krisztián Miskó, Gary Mogyorodi, Ebbe Munk, Maysinee Nakmanee, Ingvar Nordström, Tal Pe’er, Kunlanan Peetijade, Sahani Pinidiya, Lukáš Piška, Nishan Portoyan, Meile Posthuma, Yasassri Ratnayake, Randall Rice, Adam Roman, Marc Rutz, Jan Sabak, Vimukthi Saranga, Sumuduni Sasikala, Gil Shekel, Radoslaw Smilgin, Péter Sótér , Helder Sousa, Richard Taylor, Benjamin Timmermans, Giancarlo Tomasig, Robert Treffny, Shun Tsunoda, Stephanie Ulrich, François Vaillancourt, Linda Vreeswijk, Carsten Weise, Marc-Florian Wendland, Paul Weymouth, Elżbieta Wiśniewska, John Young, Claude Zhang.

Edición 2021-2022 (v3.1): Gery Ágnecz, Armin Born, Chenyifan, Klaudia Dussa-Zieger, Chen Geng (Kevin), Istvan Gercsák, Richard Green, Ole Chr. Hansen, Zsolt Hargitai, Andreas Hetz, Tobias Horn, Joan Killeen, Attila Kovacs, Rik Marselis, Marton Matyas, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Pe’er, Palma Polyak, Nishan Portoyan, Meile Posthuma, Stuart Reid, Murian Song, Péter Sótér, Lucjan Stapp, Benjamin Timmermans, Chris van Bael, Stephanie van Dijck, Paul Weymouth.

Subsequently, Tal Pe’er, Stuart Reid, Marc-Florian Wendland, and Matthias Hamburg suggested formal, grammar, and wording improvements that have been implemented and published in Errata 3.1.1 and 3.1.2.

Edición 2019 (v3.0): Laura Albert, Markus Beck, Henriett Braunné Bokor, Francisca Cano Ortiz, Guo Chaonian, Wim Decoutere, Milena Donato, Klaudia Dussa-Zieger, Melinda Eckrich-Brajer, Péter Földházi Jr, David Frei, Chen Geng, Matthias Hamburg, Zsolt Hargitai, Zhai Hongbao, Tobias Horn, Ágota Horváth, Beata Karpinska, Attila Kovács, József Kreisz, Dietrich Leimsner, Ren Liang, Claire Lohr, Ramit Manohar Kaul, Rik Marselis, Marton Matyas, Don Mills, Blair Mo, Gary Mogyorodi, Ingvar Nordström, Tal Peer, Pálma Polyák, Meile Posthuma, Lloyd Roden, Adam Roman, Abhishek Sharma, Péter Sótér, Lucjan Stapp, Andrea Szabó, Jan te Kock, Benjamin Timmermans, Chris Van Bael, Erik van Veenendaal, Jan Versmissen, Carsten Weise, Robert Werkhoven, Paul Weymouth.

Edición 2012 (v2.0): Graham Bath, Arne Becher, Rex Black, Piet de Roo, Frans Dijkman, Mats Grindal, Kobi Halperin, Bernard Homès, Maria Jönsson, Junfei Ma, Eli Margolin, Rik Marselis, Don Mills, Gary Mogyorodi, Stefan Mohacsi, Reto Mueller, Thomas Mueller, Ingvar Nordstrom, Tal Pe’er, Raluca Madalina Popescu, Stuart Reid, Jan Sabak, Hans Schaefer, Marco Sogliani, Yaron Tsubery, Hans Weiberg, Paul Weymouth, Chris van Bael, Jurian van der Laar, Stephanie van Dijk, Erik van Veenendaal, Wenqiang Zheng, Debi Zylbermann.

Notas de la versión en idioma español

Este "Programa de Estudio de Probador Certificado del ISTQB®" de Nivel Avanzado, Analista de Prueba, Versión 4.0" ha sido traducido por Spanish Software Testing Qualifications Board (SSTQB).

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)

El Comité Ejecutivo del SSTQB agradece toda aportación que permita mejorar esta traducción del programa de estudio.

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, 21 de mayo de 2025

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 Avanzado, Analista de Prueba. 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.

0.2 El probador certificado nivel avanzado analista de prueba en la prueba de software

La certificación ISTQB® analista de prueba de nivel avanzado (CTAL-TA) proporciona las habilidades necesarias para realizar la prueba de software estructurada y en profundidad a lo largo de todo el ciclo de vida del desarrollo de software. Detalla el rol y las responsabilidades del analista de prueba en cada paso de un proceso de prueba estándar y amplía importantes técnicas de prueba. La certificación de analista de prueba de nivel avanzado está dirigida a personas que posean una certificación de nivel básico y deseen seguir desarrollando sus conocimientos en análisis de prueba y técnicas de prueba.

En este programa de estudio, el analista de pruebas se entiende como un rol que:

- se concentra más en las necesidades de negocio del cliente que en los aspectos técnicos de la prueba.
- realiza, sobre todo, pruebas funcionales, pero también contribuye a pruebas no funcionales centradas en el usuario, como pruebas de usabilidad, adaptabilidad, instalabilidad o interoperabilidad.
- utiliza técnicas de prueba de caja negra y pruebas basadas en la experiencia en lugar de técnicas de prueba de caja blanca.
- mejora la efectividad de la prueba utilizando técnicas de prevención de defectos.

0.3 Trayectoria profesional de los probadores

El esquema ISTQB® apoya a los profesionales de prueba en todas las etapas de sus carreras. Las personas que obtienen la certificación de Analista de Prueba de ISTQB® de Nivel Avanzado también pueden estar interesadas en los otros Niveles Avanzados Principales (Analista de Prueba Técnica y Gestión de Prueba) y, posteriormente, en el Nivel Experto (Gestión de Prueba o Mejora del proceso de prueba). Cualquiera que busque desarrollar habilidades en prácticas de prueba en un área de entorno Ágil podría tener en cuenta las certificaciones de Probador Técnico Ágil o Liderazgo de Prueba Ágil a Escala. Además, las corrientes especializadas ofrecen productos de certificación que se concentran en tecnologías y enfoques de prueba específicos, características de calidad y niveles de prueba concretos, o pruebas dentro de dominios industriales particulares. Visite www.istqb.org para obtener la información más reciente sobre el esquema de probador certificado de ISTQB®.

0.4 Resultados de negocio

En esta sección se enumeran los resultados de negocio que se esperan de un candidato que haya obtenido la certificación de analista de prueba de nivel avanzado.

Un analista de prueba de nivel avanzado certificado puede. . .

TA-B01	Apoyar y llevar a cabo la prueba adecuada en función del ciclo de vida de desarrollo del software seguido
TA-B02	Aplicar los principios de la prueba basada en el riesgo.
TA-B03	Seleccionar y aplicar técnicas de prueba apropiadas para apoyar la consecución de los objetivos de prueba
TA-B04	Proporcionar documentación con los niveles adecuados de detalle y calidad
TA-B05	Determinar los tipos adecuados de pruebas funcionales que se deben realizar
TA-B06	Contribuir a las pruebas no funcionales
TA-B07	Contribuir a la prevención de defectos
TA-B08	Mejorar la eficiencia del proceso de prueba con el uso de herramientas
TA-B09	Especificar los requisitos de los entornos de prueba y los datos de prueba

0.5 Objetivos de aprendizaje evaluables y nivel cognitivo de conocimiento

Los objetivos de aprendizaje sustentan los resultados del negocio y se utilizan para crear los exámenes de probador certificado de nivel avanzado, analista de prueba.

En general, todos los contenidos de este programa de estudio son evaluables, excepto la Introducción y los Apéndices. Las preguntas del examen confirmarán el conocimiento de las palabras clave del nivel K1 (véase más abajo) o de los objetivos de aprendizaje del nivel de conocimientos correspondiente. Los objetivos de aprendizaje específicos y sus niveles de conocimiento se muestran al principio de cada capítulo y se clasifican de la siguiente manera:

- K2: Comprender
- K3: Aplicar
- K4: Analizar

Para todos los términos que figuran como palabras clave justo debajo de los títulos de los capítulos, se recordará el nombre y la definición correctos del glosario ISTQB® (K1), aunque no se mencionen explícitamente en ningún objetivo de aprendizaje.

En la sección 7 se ofrecen más detalles y ejemplos de objetivos de aprendizaje.

0.6 El examen del certificado de nivel avanzado de analista de prueba

El examen de certificación de analista de prueba de nivel avanzado se basará en este programa de estudio. Las respuestas a las preguntas del examen pueden requerir el uso de material basado en más de una sección de este programa de estudio. Todas las secciones del programa de estudio son evaluables, excepto la Introducción y los Apéndices. Se incluyen normas y libros como referencias, pero su contenido no es evaluable más allá de lo que se resume en el propio programa de estudio a partir de dichas normas y libros.

Consulte el documento Estructuras y normas de examen compatibles con el Analista de prueba de nivel avanzado v4.0 para obtener más detalles.

El criterio de acceso para presentarse al examen Probador Certificado de ISTQB® de Nivel Avanzado, Analista de Prueba ("ISTQB® Certified Tester Advanced Level Test Analyst") es que los candidatos estén interesados en la prueba de software. Sin embargo, se recomienda encarecidamente que los candidatos también:

- Dispongan de una formación mínima en desarrollo o prueba de software, por ejemplo, seis meses de experiencia como probador de sistemas o de aceptación de usuario o como desarrollador de software.
- Realicen un curso que haya sido acreditado según los estándares ISTQB® (por uno de los comités miembro reconocidos por ISTQB).

Nota sobre los requisitos de acceso: Se deberá obtener el certificado de nivel básico ISTQB® antes de presentarse al examen de certificación de analista de pruebas de nivel avanzado.

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 o del organismo que realice la acreditación. Un curso acreditado es reconocido como conforme a este programa de estudio y permite que un examen ISTQB® sea incluido en el curso. Las directrices de acreditación para este programa de estudio siguen las directrices generales de acreditación publicadas por el Grupo de Trabajo de Gestión de Procesos y Conformidad de ISTQB®.

0.8 Tratamiento de los estándares

Organizaciones internacionales de estandarización como IEEE e ISO han emitido estándares asociados con características de calidad y prueba de software. Tales estándares están referenciados en este programa de estudio. El propósito de estas referencias es proporcionar un marco de trabajo (como en las referencias a ISO/IEC 25010 respecto a las características de calidad) o proporcionar una fuente de información adicional si así lo desea el lector. Se debe tener en cuenta que los programas de estudio ISTQB® utilizan documentos estándar como referencia. Los documentos estándar no están destinados a ser evaluables. Para más información sobre los estándares, consultar la sección 6 - Referencias.

0.9 Nivel de detalle

El nivel de detalle de este programa de estudio permite que los cursos y los exámenes sean consistentes a nivel internacional. Para lograr este objetivo, el programa de estudio consta de:

- Objetivos generales de enseñanza que describen la intención del analista de prueba de nivel avanzado.
- Una lista de términos que los alumnos deben ser capaces de recordar.
- Objetivos de aprendizaje para cada área de conocimiento, que describen los resultados cognitivos del aprendizaje que deben alcanzarse.
- Una descripción de los conceptos clave, incluyendo referencias a fuentes como la bibliografía o los estándares aceptados.

El contenido del programa de estudio no describe toda el área de conocimiento de la prueba de software; refleja el nivel de detalle que se cubrirá en los cursos de formación de analista de prueba de nivel avanzado.

El programa de estudio utiliza la terminología (es decir, el nombre y el significado) de los términos empleados en la prueba y el aseguramiento de la calidad del software de acuerdo con el Glosario ISTQB® (ISTQB-Glossary, 2024).

Para la terminología en disciplinas relacionadas, consulte los glosarios respectivos: IREB-CPRE para la ingeniería de requisitos (IREB-Glossary, 2024), e IEEE-Pascal para la ingeniería de software (Computer Society et al., 2024).

0.10 Organización del programa de estudio

Hay cinco capítulos con contenidos evaluables. El encabezamiento superior de cada capítulo especifica el tiempo asignado al 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 20,25 horas lectivas (1215 minutos), distribuidas en los cinco capítulos del siguiente modo:

- **Capítulo 1** (225 minutos): Las tareas del analista de prueba en el proceso de prueba
  - El alumno aprende cómo participa el analista de prueba en los distintos ciclos de vida del desarrollo del software.
  - El alumno aprende cómo participa el analista de prueba en diversas actividades de prueba.
  - El alumno aprende las tareas que realiza el analista de prueba en relación con los productos de trabajo.
- **Capítulo 2** (90 minutos) Las tareas del analista de prueba en la prueba basada en el riesgo
  - El alumno aprende cómo contribuye el analista de prueba al análisis del riesgo de producto.
  - El alumno aprende a analizar el impacto de los cambios para determinar el alcance de la prueba de regresión.
- **Capítulo 3 (**615 minutos) Análisis de prueba y diseño de prueba
  - El alumno aprende acerca de las técnicas de prueba basadas en datos, como la prueba de dominio, la prueba combinatoria y la prueba aleatoria.
  - El alumno aprenderá las técnicas de prueba basadas en el comportamiento, como la prueba CLAB (por acrónimo en inglés «CRUD»), la prueba de transición de estado y la prueba de escenario.
  - El alumno aprende técnicas de prueba basadas en reglas, como la prueba de tabla de decisión y la prueba metamórfica.
  - El alumno aprende sobre las pruebas basadas en la experiencia, como las pruebas basadas en la sesión y las pruebas multitudinarias.
  - El alumno aprende a seleccionar las técnicas de prueba adecuadas para mitigar los riesgos de producto.
- **Capítulo 4** (60 minutos) Prueba de las características de calidad
  - El alumno aprende a realizar varios tipos de pruebas funcionales.
  - El alumno aprende a utilizar el conocimiento específico de la funcionalidad para contribuir a los tipos de pruebas no funcionales, como las pruebas de usabilidad, las pruebas de flexibilidad y las pruebas de compatibilidad.
- **Capítulo 5** (225 minutos) Prevención de defectos en el software
  - El alumno aprende sobre diversas prácticas de prevención de defectos.
  - El alumno aprende sobre varios enfoques que apoyan la contención de fase.
  - El alumno aprende a mitigar la recurrencia de defectos.

1 Tareas del analista de prueba en el proceso de prueba

Duración: 225 minutos

Palabras Clave[1]

En la siguiente tabla se presentan las palabras clave del capítulo. En este documento, se identifican dos tipos de palabras clave:

- **ISTQB**: identificarán palabras clave del proceso de prueba
- **ESPDOM**: identificarán palabras clave específicas de dominio

Tipo Palabra Clave	Español	Inglés
ISTQB	caso de prueba de alto nivel	high-level test case
ISTQB	palabra clave	keyword
ISTQB	prueba guiada por palabra clave	keyword-driven testing
ISTQB	caso de prueba de bajo nivel	low-level test case
ISTQB	ciclo de vida de desarrollo del software	software development lifecycle
ISTQB	análisis de prueba	test analysis
ISTQB	analista de prueba	test analyst
ISTQB	caso de prueba	test case
ISTQB	condición de prueba	test condition
ISTQB	datos de prueba	test data
ISTQB	diseño de prueba	test design
ISTQB	entorno de prueba	test environment
ISTQB	ejecución de pruebas	test execution
ISTQB	implementación de prueba	test implementation
ISTQB	oráculo de prueba	test oracle
ISTQB	guion de prueba	test script
ISTQB	producto de prueba	testware

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

Objetivos de Aprendizaje para "Capítulo 1"

1.1 La prueba en el ciclo de vida del desarrollo de software

TA-1.1.1 **(K2)** Resumir la participación del analista de prueba en diversos ciclos de vida del desarrollo de software.

1.2 Participación en actividades de prueba

TA-1.2.1 **(K2)** Resumir las tareas realizadas por el analista de prueba como parte del análisis de prueba.

TA-1.2.2 **(K2)** Resumir las tareas realizadas por el analista de prueba como parte del diseño de prueba

TA-1.2.3 **(K2)** Resumir las tareas realizadas por el analista de prueba como parte de la implementación de prueba.

TA-1.2.4 **(K2)** Resumir las tareas realizadas por el analista de prueba como parte de la ejecución de la prueba.

1.3 Tareas relacionadas con productos de trabajo

TA-1.3.1 **(K2)** Diferenciar entre casos de prueba de alto nivel y casos de prueba de bajo nivel.

TA-1.3.2 **(K2)** Explicar los criterios de calidad para casos de prueba.

TA-1.3.3 **(K2)** Aportar ejemplos de requisitos de entorno de prueba.

TA-1.3.4 **(K2)** Explicar el problema del oráculo de prueba y las posibles soluciones.

TA-1.3.5 **(K2)** Aportar ejemplos de requisitos de datos de prueba.

TA-1.3.6 **(K3)** Utilizar pruebas guiadas por palabras clave para desarrollar guiones de prueba.

TA-1.3.7 **(K2)** Resumir los tipos de herramientas para gestionar los productos de prueba.

Introducción a las tareas del analista de prueba en el proceso de prueba

El programa de estudio de nivel básico (ISTQB-CTFL, v4.0.1) describe dos roles principales en las pruebas: un rol de gestión de la prueba y un rol de prueba. En este programa de estudio, la persona con un rol de prueba responsable de probar los aspectos de negocio del software se denomina analista de prueba (AP). Aunque esta responsabilidad rara vez se asigna a un puesto o rol específico, el AP cuenta con tareas claramente definidas y competencias necesarias. En cuanto a los niveles de prueba, el AP se concentra en la prueba de sistema, la prueba de aceptación y la prueba de integración de sistemas. En cuanto a las características de calidad, las competencias de la AP se concentran en la adecuación funcional y también abarcan ciertas características de calidad no funcionales de cara al usuario, como la usabilidad, la adaptabilidad, la instalabilidad y la interoperabilidad.

1.1 La prueba en el ciclo de vida del desarrollo de software

1.1.1 Participación del analista de prueba en diversos ciclos de vida de desarrollo del software

La organización de las actividades de prueba puede variar en función del ciclo de vida de desarrollo de software (CVDS) que se siga. Por lo tanto, la implicación del AP en las actividades de prueba puede variar en función del modelo de ciclo de vida de desarrollo de software (CVDS) adoptado.

En los **modelos de desarrollo secuencial**, las actividades de desarrollo se realizan por fases y comienzan cuando se completa la fase anterior. Normalmente, hay poco solapamiento entre estas actividades. Por lo tanto, las tareas del AP suelen cambiar con el tiempo. En las primeras fases del ciclo de vida de desarrollo de software (CVDS), el AP se concentra en apoyar la planificación de la prueba. El AP comienza el análisis de prueba cuando se está elaborando la base de prueba. El diseño de prueba y la implementación de prueba siguen en paralelo con el diseño y la implementación del software. Por último, el AP ejecuta las pruebas y apoya la compleción de la prueba durante las fases finales del ciclo de vida de desarrollo de software (CVDS).

Los modelos de desarrollo incremental dividen el software en incrementos más pequeños y manejables. Cada incremento se desarrolla y prueba de forma independiente. Por lo tanto el AP realiza las mismas actividades para cada incremento (es decir, análisis de prueba, diseño de prueba, implementación de prueba, ejecución de prueba y apoyo a la gestión de prueba). Sin embargo, el trabajo del AP puede organizarse de forma diferente para cada incremento. La prueba se concentra en las prestaciones nuevas o modificadas. Además, debido al mayor riesgo de regresión, el AP debe prestar especial atención a la refactorización y al montaje de conjuntos de pruebas de regresión.

En **los modelos de desarrollo iterativo**, el proceso de desarrollo es cíclico. El proyecto se somete a repetidos ciclos de creación de prototipos, prueba, refinamiento y despliegue. El rol del AP es dinámico y adaptativo. El AP colabora estrechamente con los desarrolladores y los representantes de negocio, adaptándose a la evolución del producto. El AP adapta y modifica las condiciones de prueba y los casos de prueba a medida que evoluciona el software y proporciona retroalimentación para mejorar el proceso de prueba en cada iteración. Cuanto más frecuentes sean las iteraciones, más críticos serán el mantenimiento y el desarrollo continuos de las pruebas de regresión por parte del AP.

Un CVDS puede combinar elementos de varios modelos y técnicas y enfoques específicos (por ejemplo, el Desarrollo Ágil de Software combina aspectos de los modelos iterativo e incremental). En estos casos, la participación del AP dependerá de las características específicas del CVDS [ciclo de vida de desarrollo de software (SDLC)] y de cómo se combinen. Una buena práctica común a todos los modelos de CVDS es que el AP debe participar desde las fases iniciales del CVDS [ciclo de vida de desarrollo de software (CVDS)].

1.2 Participación en actividades de prueba

El programa de estudio de nivel básico (ISTQB-CTFL, v4.0.1) describe siete actividades dentro del proceso de prueba. El AP se concentra principalmente en cuatro de ellas: análisis de prueba, diseño de prueba, implementación de prueba y ejecución de prueba.

Las tareas del AP en estas cuatro actividades se describen con detalle en las siguientes secciones.

1.2.1 Análisis de prueba

Durante el análisis de prueba, el AP comprueba la completitud de la base de prueba y recoge cualquier información adicional relevante para la prueba. Esto incluye no sólo documentación, sino también información verbal, por ejemplo, conversaciones en la redacción colaborativa de historias de usuario (véase ISTQB-CTFL (v4.0.1), sección 4.5.1). Los cambios en la base de prueba pueden dar lugar a ajustes en el alcance de la prueba en coordinación con la gestión de la prueba.

Durante el análisis de prueba, el AP comprueba la completitud de la base de prueba y recoge cualquier información adicional relevante para la prueba. Esto incluye no sólo documentación, sino también información verbal, por ejemplo, conversaciones en la redacción colaborativa de historias de usuario (véase ISTQB-CTFL (v4.0.1), sección 4.5.1). Los cambios en la base de prueba pueden dar lugar a ajustes en el alcance de la prueba en coordinación con la gestión de la prueba.

Para proceder eficazmente al análisis de prueba, el AP comprueba los siguientes criterios de entrada:

- Se ha realizado la planificación de la prueba y están claros el alcance, los objetivos y el enfoque de prueba.
- La base de prueba (que contiene información como requisitos o historias de usuario) está definida.
- Los riesgos de producto ya identificados se han evaluado y documentado si fuera necesario.

El AP evalúa la base de prueba para identificar cualquier defecto que pudiera contener y evaluar su capacidad de ser probado, proporcionando así una retroalimentación temprana a los propietarios de producto. Esto puede incluir el modelado del comportamiento del sistema según las técnicas de prueba que se vayan a aplicar (véase la Sección 3, Sección 5.2.1). También se aplican técnicas de revisión como parte del proceso (véase la Sección 5.2.2). Si no se solucionan directamente, los defectos relativos a la base de prueba deben documentarse. Además, el AP determina los oráculos de prueba necesarios (véase la sección 1.3.4).

El AP define y prioriza las condiciones de prueba para cada elemento de prueba del alcance. Las condiciones de prueba abordan los objetivos de prueba (véase ISTQB-CTFL (v4.0.1), Sección 1.1.1) y deben ser trazables a los elementos de la base de prueba. El alcance y el enfoque de las condiciones de prueba tienen en cuenta los riesgos del producto. En los modelos de desarrollo incremental o iterativo, esto incluye determinar el alcance de las pruebas de regresión basándose en un análisis de impacto. En el Desarrollo Ágil de Software, las condiciones de prueba pueden expresarse como criterios de aceptación que reflejen los riesgos de las historias de usuario.

El AP puede proceder por etapas, empezando por condiciones de prueba de alto nivel como “funcionalidad de la pantalla x”. A continuación, el AP define condiciones de prueba más detalladas como “la pantalla x rechaza los números de cuenta que tienen un dígito menos”. Este enfoque favorece una cobertura suficiente y permite empezar pronto con el diseño de la prueba, por ejemplo, para las historias de usuario que aún necesitan refinarse.

El AP involucra a los implicados en la revisión de las condiciones de prueba para asegurar que la base de prueba se entiende claramente y que la prueba está alineada con los objetivos de prueba.

1.2.2 Diseño de prueba

El diseño de prueba describe cómo realizar las pruebas para alcanzar los objetivos de prueba establecidos. Esto se suele hacer con casos de prueba. La forma en que se realiza el diseño de la prueba depende de muchos factores, como la cobertura necesaria, la base de prueba, el ciclo de vida de desarrollo de software (CVDS), las restricciones del proyecto y los conocimientos y la experiencia de los probadores.

Durante el diseño de pruebas, el AP determina en qué áreas resultan adecuados los casos de prueba de bajo nivel o los casos de prueba de alto nivel (véase la sección 1.3.1). En ambos casos, el AP debe identificar criterios claros de paso/fallo. El AP diseña los casos de prueba para las condiciones de prueba nuevas o modificadas de acuerdo con los criterios de calidad (véase la Sección 1.3.2). Para las pruebas de regresión, suele ser suficiente la selección de casos de prueba de alto nivel existentes o la adaptación de casos de prueba de bajo nivel existentes en función de su priorización.

El AP capta la trazabilidad entre la base de prueba, las condiciones de prueba y los casos de prueba. En la prueba basada en la experiencia, los casos de prueba no siempre están documentados; en su lugar, las condiciones de prueba (entre otras) pueden guiar la ejecución de la prueba. El AP puede diseñar algunos casos de prueba basándose en objetivos de prueba de alto nivel.

Además de estas tareas el AP define los requisitos del entorno de prueba (véase el apartado 1.3.3) e identifica, crea y especifica los requisitos de los datos de prueba (véase el apartado 1.3.5).

El AP utiliza los criterios de salida definidos en la planificación de la prueba para determinar cuándo se han diseñado suficientes casos de prueba. Sin embargo, otros criterios de salida como los niveles de riesgo residual o las restricciones del proyecto (por ejemplo, el presupuesto o el tiempo) también indican cuándo puede terminar el diseño de prueba.

El diseño de prueba puede apoyarse en herramientas, pero debería ser agnóstico en cuanto a herramientas y tecnología para seguir siendo independiente de ellas. El diseño de prueba aplica un enfoque sistemático utilizando técnicas de prueba o es ad hoc en caso contrario.

Los casos de prueba tienen un rol comunicativo y deben ser comprensibles para los implicados relevantes. Como un caso de prueba no siempre puede ser ejecutado por su autor, otros probadores necesitan entender cómo ejecutarlo, sus objetivos de prueba (es decir, sus condiciones de prueba subyacentes) y su importancia. Los casos de prueba también deben ser comprensibles para los desarrolladores, que pueden implementar las pruebas o volver a ejecutarlas en caso de fallo, y para los auditores, que puedan tener que aprobarlas.

1.2.3 Implementación de prueba

Durante la implementación de prueba, el AP proporciona el producto de prueba que se necesita para la ejecución de prueba. El AP puede organizar los procedimientos de prueba y los guiones de prueba en conjuntos de prueba o sugerir casos de prueba para su automatización. Definir los procedimientos de prueba requiere identificar cuidadosamente las restricciones y dependencias que pueden influir en el orden de ejecución de la prueba. Además de los pasos contenidos en los casos de prueba, los procedimientos de prueba incluyen pasos para establecer cualquier precondición inicial (por ejemplo, cargar los datos de prueba de un repositorio de datos), verificar los resultados esperados y las postcondiciones, y restablecer los pasos posteriores a la ejecución (por ejemplo, restablecer la base de datos, el entorno y el sistema).

El AP prioriza los procedimientos de prueba y guiones de prueba para la ejecución de la prueba basándose en los criterios de priorización identificados durante el análisis del riesgo y la planificación de la prueba e identifica los procedimientos de prueba o guiones de prueba que deben ejecutarse en la versión actual del objeto de prueba. Esto permite que las pruebas relacionadas (por ejemplo, para nuevas prestaciones o pruebas de regresión) se ejecuten juntas en una realización de prueba específica. El AP actualiza la

trazabilidad entre la base de prueba y otro producto de prueba, como los procedimientos de prueba, los guiones de prueba y los juegos de prueba.

El AP puede ayudar a la gestión de la prueba (GP) a establecer un calendario de ejecución de prueba, incluida la asignación de recursos, para permitir una ejecución de prueba eficiente mediante la definición del orden de ejecución de la prueba (véase ISTQB-CTFL, v4.0.1, sección 5.1.5).

El AP crea datos de entrada y de entorno para cargarlos en bases de datos y otros repositorios (véase la sección 1.3.5). Estos datos deben ser “aptos para el propósito” para apoyar los objetivos de prueba específicos.

El AP también debe verificar que el entorno de prueba se encuentre totalmente configurado y listo para la ejecución de prueba (véase la Sección 1.3.3). La mejor forma de hacerlo es diseñando y ejecutando una prueba de humo. El entorno de prueba debe revelar los defectos del objeto de prueba mediante la ejecución de la prueba, funcionar con normalidad cuando no se produzcan fallos y reproducir adecuadamente, si es necesario, el entorno de producción o del usuario final.

El nivel de detalle y la complejidad asociada del trabajo realizado durante la implementación de prueba pueden verse influidos por el nivel de detalle de las condiciones de prueba y los casos de prueba. En algunos casos, se aplican normas reglamentarias, y el producto de prueba debe proporcionar evidencia de conformidad con los estándares aplicables (por ejemplo, RTCA DO- 178C, 2011).

1.2.4 Ejecución de pruebas

La ejecución de pruebas se lleva a cabo de acuerdo con el calendario de ejecución de prueba. Las tareas típicas que realiza el AP son la ejecución de pruebas, la comparación de los resultados reales con los resultados esperados, el análisis de anomalías, el suministro de información sobre defectos y el registro de los resultados de las pruebas.

El AP ejecuta pruebas de forma manual. Esto incluye pruebas exploratorias, ejecución de procedimientos de prueba, pruebas de regresión y pruebas de confirmación. Para las pruebas exploratorias, el AP puede utilizar pruebas basadas en sesiones con contratos de prueba (véase el apartado 3.4.1). El AP también puede ejecutar guiones de prueba automatizados, pero puede ser tarea de los desarrolladores, de los ingenieros de automatización de la prueba (IAP) o de los analistas de prueba técnica (APT) ejecutar guiones de prueba automatizados y analizarlos en caso de fallos.

El AP analiza las anomalías que se producen en las ejecuciones manuales o automatizadas de la prueba para establecer sus causas probables. Una anomalía puede ser consecuencia de un defecto en un objeto de prueba. Sin embargo, también pueden existir otras razones, como la falta de precondiciones, datos de prueba incorrectos, defectos en los guiones de prueba o en el entorno de prueba, o malentendidos en las especificaciones. El AP registra los resultados reales de la ejecución de prueba, comunica los defectos basándose en los fallos observados e informa sobre ellos si fuera necesario.

El AP actualiza la trazabilidad entre la base de prueba y otros productos de prueba teniendo en cuenta los resultados de las pruebas. Esta información permite transformar los resultados de prueba en información de alto nivel sobre riesgos o cobertura, lo que permite a los implicados tomar decisiones con conocimiento de causa. Por ejemplo, esto aclara a los implicados cuántos casos de prueba relacionados con una condición de prueba han pasado o fallado.

Además de estas tareas típicas, el AP evalúa los resultados de prueba, incluyendo las siguientes tareas:

- Reconocimiento de agrupamientos de defectos, que pueden indicar la necesidad de probar más una parte concreta del objeto de prueba (véase la sección 5.3.1).
- Volver a ejecutar, de forma manual, pruebas automatizadas que hayan fallado para asegurarse de que la automatización de la prueba no produjo un resultado de falso positivo.
- Sugerir pruebas adicionales basadas en lo aprendido durante las pruebas anteriores.
- Identificando nuevos riesgos a partir de la información obtenida al realizar la ejecución de pruebas.
- Sugerir mejoras en el diseño de pruebas o en su implementación (por ejemplo, mejoras en los procedimientos de prueba) o incluso en el sistema sujeto a prueba.
- Sugerir mejoras en los juegos de prueba de regresión, como la refactorización, los ajustes de alcance y la automatización de la prueba (véase la sección 2.2.1).

1.3 Tareas relacionadas con productos de trabajo

El AP debe asegurar la calidad de los productos de trabajo de los que es responsable. Esto incluye casos de prueba, entornos de prueba, datos de prueba, oráculos de prueba y guiones de prueba. En esta sección, el programa de estudio analiza las tareas del AP relacionadas con el producto de prueba y los tipos de herramientas para gestionar el producto de prueba.

1.3.1 Casos de prueba de alto nivel y casos de prueba de bajo nivel

Un caso de prueba de alto nivel (también llamado caso de prueba abstracto o caso de prueba lógico) describe las circunstancias en las que se examina el objeto de prueba indicando qué condiciones de prueba cubre el caso de prueba. Los casos de prueba de alto nivel son, por tanto, adecuados para asegurar que las pruebas cubren todas las condiciones de prueba relevantes. Los casos de prueba de alto nivel no contienen información concreta sobre las precondiciones, los datos de entrada, las salidas previstas o las postcondiciones. Todos ellos se expresan a un nivel abstracto (por ejemplo, “pedir más de un libro, con el precio del pedido resultante en un descuento; resultado esperado: se asigna el descuento”).

Un caso de prueba de bajo nivel (también llamado caso de prueba concreto o caso de prueba físico) es el refinamiento detallado de un caso de prueba de alto nivel. Los casos de prueba de bajo nivel describen qué datos hay que preparar, qué acciones debe realizar el probador (si es necesario) y cuál es el resultado esperado concreto. Los casos de prueba de bajo nivel contienen condiciones previas específicas, datos de entrada, resultados esperados y condiciones posteriores (por ejemplo, “pedir los libros B1 (10 U.M.) y B2 (20 U.M.), con un precio total de pedido de 30 U.M.; resultado esperado: 10% de descuento asignado, precio total: 27 U.M.”).

Normalmente, un AP diseña inicialmente casos de prueba de alto nivel, que son la base para desarrollar casos de prueba de bajo nivel. Un caso de prueba de alto nivel puede implementarse en uno o más casos de prueba de bajo nivel. A veces, el caso de prueba puede seguir siendo de alto nivel, con la información concreta determinada por el AP durante la ejecución de la prueba.

Por ejemplo, los casos de prueba de alto nivel pueden guiar al AP a la hora de crear objetivos de prueba dentro de un contrato de prueba para la prueba basada en la sesión, permitiendo a los AP profundizar en estos objetivos durante la ejecución de la prueba (véase el apartado 3.4.1).

La transición de los casos de prueba de alto nivel a los de bajo nivel es algo más que rellenar valores concretos. Es también un paso de lo conceptual a lo técnico. A menudo se posterga del diseño de prueba a la implementación de prueba, especialmente si se necesitan datos de prueba específicos. El AP debe

asegurar que se conoce todo lo necesario para ejecutar los casos de prueba de bajo nivel (Koomen et al., 2006). En la práctica, muchos casos de prueba son híbridos, siendo concretos en algunos aspectos y abstractos en otros. Esto suele deberse a un compromiso entre la mantenibilidad y la comprensibilidad de los casos de prueba.

1.3.2 Criterios de calidad para los casos de prueba

Descuidar la calidad de los casos de prueba puede acarrear muchos problemas, como altos costes de mantenibilidad, comprensibilidad reducida o retrasos en la ejecución. Los criterios de calidad para los casos de prueba son un primer paso hacia casos de prueba más mantenibles. Estos criterios incluyen:

- **Corrección.** Un caso de prueba debe facilitar la verificación exacta de las condiciones de prueba en las que se basa.
- **Viabilidad.** Debe ser posible ejecutar un caso de prueba.
- **Necesidad.** Cada caso de prueba debe cubrir un objetivo de prueba claro, expresado en su título o resumen. Deben evitarse los duplicados. Las cosas que no deben probarse no deben tener casos de prueba diseñados.
- **Capacidad de ser entendido.** Los casos de prueba pueden ser revisados, modificados y ejecutados por personas distintas del autor. El AP debe redactar los casos de prueba en un lenguaje y un formato comprensibles para todos los implicados, sin explicar lo obvio. Los casos de prueba complejos deben simplificarse o dividirse.
- **Trazabilidad.** Los casos de prueba deben ser trazables a las condiciones de prueba, requisitos y riesgos para permitir al AP mantenerlos actualizados (véase ISTQB-CTFL (v4.0.1), sección 1.4.4).
- **Consistencia.** La consistencia en el lenguaje, el formato y la estructura facilita la comprensión y el mantenimiento de los casos de prueba. El AP puede utilizar un glosario para este fin.
- **Precisión.** Sólo debe haber una interpretación de un caso de prueba para evitar resultados de falso negativo y falso positivo. Deben evitarse términos ambiguos como “adecuado”, “según sea necesario” o “varios”.
- **Completitud.** Deben estar presentes todos los atributos necesarios (por ejemplo, véase ISO/IEC/IEEE 29119-3, 2021), incluidos los datos de prueba requeridos (véase la sección 1.3.5) y un resultado esperado claro para evitar dudas al comparar con el resultado real.
- **Concisión.** La granularidad de los casos de prueba (es decir, un caso de prueba grande con muchas acciones de prueba frente a varios casos de prueba más pequeños) debe corresponderse con la base de prueba y las condiciones de prueba. Son preferibles los casos de prueba más pequeños concentrados en unos pocos elementos de cobertura, ya que facilitan el hallazgo de las causas de los fallos, pueden combinarse con flexibilidad en procedimientos de prueba y conjuntos de pruebas, y un fallo durante la ejecución de la prueba no bloquearía ninguna otra prueba.

El formato y el nivel de detalle de los casos de prueba dependen del contexto del proyecto y del producto y deben acordarse dentro del equipo de prueba.

1.3.3 Requisitos del entorno de prueba

El entorno de prueba es un factor crítico de éxito tanto para la ejecución de prueba manual como para la automatizada. La implementación del entorno de prueba influye en la capacidad de ser probado, la detección de defectos, los costes generales de las pruebas y la fiabilidad de los resultados de las pruebas. Un caso de prueba que pasa o falla en el entorno de prueba debe tener el mismo resultado de prueba cuando se ejecuta en producción. Lo ideal es un entorno de prueba robusto, predecible e integrado con el

marco de automatización de la prueba, si es necesario. El AP puede definir los requisitos del entorno de prueba durante el diseño de prueba basándose en el análisis de:

- condiciones de prueba, casos de prueba y requisitos de datos de prueba, de forma que los requisitos del entorno de prueba describan las condiciones necesarias para configurar y mantener el entorno de prueba a fin de asegurar que se cumplen las condiciones previas de la ejecución de prueba.
- niveles de prueba y tipos de prueba, que influyen en el equilibrio entre la flexibilidad del entorno de prueba y la similitud con el entorno de producción.
- disponibilidad e independencia de los componentes y sistemas, que pueden indicar la necesidad de utilizar dobles de prueba (por ejemplo, stubs o controladores).

Los requisitos del entorno de prueba describen los elementos del entorno de prueba, que pueden clasificarse en varios tipos, como hardware, middleware, software, servicios virtualizados, red, interfaces, herramientas, seguridad, configuración y lugar. Para cada elemento de entorno de prueba, los requisitos deben incluir la siguiente información (ISO/IEC/IEEE 29119-3, 2021):

- **identificador único:** utilizado con fines de trazabilidad
- **descripción:** con suficiente detalle para implementarlo según sea necesario
- **responsabilidad:** describe quién es responsable de hacer que esté disponible
- **período necesario:** identifica cuándo y durante cuánto tiempo se necesita el elemento
- **fidelidad:** el grado en que este elemento representa o se desvía del entorno de producción

Los requisitos también deben abordar las necesidades generales del entorno de prueba, incluyendo la configuración, la copia de seguridad y la restauración, las necesidades de seguridad, la capacidad de cambiar el entorno de prueba y los roles y autorizaciones (Koomen et al., 2006).

El AP debe documentar los requisitos del entorno de prueba de forma clara, concisa y coherente. El AP puede utilizar diagramas o tablas. Para evitar redundancias y documentación innecesaria, el AP puede hacer referencia a entornos de prueba existentes y centrarse en las necesidades específicas del nivel de prueba. Los implicados relevantes (por ejemplo, desarrolladores, TTA, ingenieros de automatización de pruebas, analistas de negocio, patrocinadores y propietarios de productos) también deben revisar, aprobar y actualizar los requisitos del entorno de prueba.

1.3.4 Determinación de los oráculos de prueba

Se necesita un oráculo de prueba para determinar los resultados esperados en las pruebas dinámicas. Lo ideal es que la base de prueba proporcione los oráculos de prueba (por ejemplo, en una especificación textual o formal). La experiencia o el conocimiento humano sobre el objeto de prueba también pueden servir como oráculo de prueba. Puede ser necesario un oráculo de prueba automatizado por razones de rentabilidad (por ejemplo, si las entradas se generan automáticamente o los oráculos humanos son costosos).

Dependiendo de la calidad y completitud de la base de prueba o de las características del sistema, es posible que no se disponga de un oráculo de prueba rentable. Esto se conoce como el problema del oráculo de prueba. Los factores típicos que contribuyen al problema del oráculo de prueba incluyen la complejidad relacionada con los datos, el no determinismo (por ejemplo, los sistemas basados en IA), el comportamiento probabilístico y los requisitos faltantes o ambiguos.

Algunas soluciones conocidas al problema del oráculo de prueba son (Barr et al., 2014):

- Los pseudooráculos son sistemas desarrollados de forma independiente que cumplen la misma especificación que el objeto de prueba (por ejemplo, sistemas legados, versiones simplificadas de objetos de prueba). El desarrollo de un pseudooráculo dedicado para un objeto de prueba complejo puede ser costoso, pero es habitual en sistemas críticos.
- La prueba basada en modelos puede formalizar el oráculo de prueba como parte del modelo de prueba. Permite generar salidas esperadas y derivar pruebas a partir del modelo. Las técnicas de prueba basada en el comportamiento suelen implicar la implementación de un oráculo de prueba automatizado (véanse las secciones 3.2.2 y 3.2.3).
- La prueba basada en propiedades utiliza propiedades especificadas del objeto de prueba para verificar las relaciones entre la entrada y el resultado esperado de casos de prueba individuales. Si no se cumple dicha relación, el caso de prueba falla. Esta solución es muy adecuada para la automatización de la prueba, pero su efectividad depende de las relaciones, que pueden ser difíciles de determinar.
- Prueba metamórfica (véase la sección 3.3.2).
- Las aserciones pueden construirse en el código de automatización de pruebas o en el propio objeto de prueba para implementar un oráculo de prueba automatizado. Son enunciados ejecutables que verifican el estado o el comportamiento del objeto de prueba. Cuando se construyen en el objeto de prueba, normalmente solo verifican lo necesario para la continuación de la tarea.

1.3.5 Requisitos de datos de prueba

Durante el diseño de la prueba, el AP identifica y solicita los datos de prueba que pueden necesitar preparación o aprovisionamiento para la ejecución de la prueba. Se tienen en cuenta aspectos como la finalidad, el formato y el contexto de uso. (ISO/IEC/IEEE 29119-3, 2021, sección 8.5) también describe los requisitos de los datos de prueba. Los aspectos clave son:

- **Similitud con los datos de producción.**
  - Los datos de producción reflejan datos del mundo real, pero pueden carecer de variabilidad. Los datos sintéticos permiten una variabilidad controlada. Deben reflejar aspectos clave de los patrones, distribuciones y valores atípicos de los datos de producción, pero pueden pasar por alto ciertos defectos que solo se producen con datos del mundo real. En el caso de los sistemas que carecen de datos de producción, los datos sintéticos deben reflejar escenarios técnicos y de negocio realistas. Las personas ayudan al proporcionar perfiles realistas y centrados en el usuario que guían la creación de datos que reflejan diversos escenarios y comportamientos de los usuarios.
- **Confidencialidad.**
  - Los datos de prueba sensibles (por ejemplo, la información personal) requieren protección. Los datos seudonimizados sustituyen la información personal por identificadores artificiales. Los datos anonimizados eliminan la información identificable sobre las personas. En caso necesario, el AP debe respetar normativas de protección de datos como la GDPR en la Unión Europea (Comisión, 2016) o la HIPAA en EE.UU. (Health et al., 2024).
- **Objetivo.**
  - Los datos de prueba son cruciales para determinar las precondiciones y los resultados esperados que afectan al estado del sistema y a su configuración (por ejemplo, la hora y

la fecha del sistema). También incluye la especificación de los permisos de usuario y el establecimiento de relaciones entre productos, departamentos y categorías.

- **Criterios de cobertura.**
  - Los datos de prueba deben ajustarse a los criterios de cobertura de la técnica de prueba elegida. Además de datos de prueba válidos, esto también puede requerir datos de prueba no válidos, como en el caso de las pruebas negativas.
- **Formato de los datos.**
  - La gestión sistemática de datos (por ejemplo, en las pruebas de API) puede requerir datos estructurados (por ejemplo, CSV, JSON, XML o bases de datos).
- **Trazabilidad.**
  - La trazabilidad asegura la mantenibilidad de los datos de prueba cuando se realizan cambios en los casos de prueba.
- **Mantenibilidad.**
  - Deben evitarse los datos de prueba codificados de forma rígida en casos de prueba de bajo nivel para facilitar la detección de defectos y el mantenimiento. Los casos de prueba deben separar la lógica de los datos de prueba (véase la sección 1.3.2).
- **Dependencias.**
  - Los datos dependientes requieren una serie de pasos para su creación.
- **Disponibilidad.**
  - La virtualización de servicios puede resolver la falta de datos simulando servicios ausentes o inaccesibles para interactuar con sistemas o servicios externos.
- **Sensibilidad temporal y envejecimiento de los datos.**
  - Los casos de prueba que incluyen datos no actualizados o sensibles al tiempo pueden afectar al comportamiento del sistema de forma inesperada o imprecisa.

1.3.6 Desarrollo de guiones de prueba mediante pruebas guiadas por palabras clave

Si se utilizan pruebas guiadas por palabras clave, el AP crea guiones de prueba utilizando palabras clave. La implementación de los guiones de prueba es tarea del TTA, el Ingeniero de Automatización de Pruebas (IAP) o un desarrollador.

El AP identifica y especifica palabras clave mediante el análisis de la base de pruebas o la colaboración con las partes interesadas. Las palabras clave se pueden clasificar en dos categorías: acción y verificación. Las palabras clave de acción deben interactuar con el objeto de prueba (por ejemplo, ejecutar funciones, enviar datos, navegar dentro del objeto de prueba), el entorno de prueba (por ejemplo, configurar y activar simuladores) u otros componentes o sistemas (por ejemplo, activar una interfaz del objeto de prueba). Las palabras clave de verificación representan aserciones para evaluar si el resultado real producido por el objeto de prueba coincide con el resultado esperado.

Las palabras clave residen en al menos dos capas de abstracción: la capa de dominio y la capa de interfaz de prueba. Las palabras clave de la capa de dominio corresponden a acciones relacionadas con el negocio y reflejan la terminología del dominio de la aplicación. Abstraen los detalles técnicos de la interfaz del objeto de prueba. Las palabras clave de la capa de interfaz de prueba se comunican con los objetos de prueba, los elementos del entorno de prueba u otros componentes o sistemas a través de su interfaz de

prueba. Residen en la capa de abstracción más baja. Las capas intermedias adicionales pueden soportar la mantenibilidad de las palabras clave.

Las palabras clave pueden ser atómicas o estar compuestas por otras palabras clave. La estructura y la capa de abstracción de una palabra clave son atributos independientes. Sin embargo, las palabras clave compuestas suelen residir en un nivel de abstracción más alto, mientras que las palabras clave atómicas tienden a residir en la capa de interfaz de prueba.

Las tareas del AP en las pruebas guiadas por palabras clave incluyen:

- Especificar palabras clave y sus parámetros.
- Especificar los casos de prueba basados en palabras clave, es decir, guiones de prueba que utilizan palabras clave.
- Especificar los pasos adicionales a los guiones de prueba utilizando palabras clave como precondiciones, acciones de verificación y limpieza del entorno de prueba después de la prueba.
- Mantener los casos de prueba basados en palabras clave para reflejar los cambios en el objeto de prueba.
- Ejecutar guiones de prueba basados en palabras clave, ya sea de forma automatizada o manual.
- Analizar los casos de prueba de palabras clave fallidos para determinar la causa del fallo.

Al analizar la base de la prueba, el AP busca interacciones entre el objeto de prueba y su entorno (por ejemplo, usuarios, otros sistemas y dispositivos). Por ejemplo, considerando la historia de usuario “Como miembro, quiero autenticarme para poder acceder a las instalaciones”, con los criterios de aceptación “Las tarjetas de miembro válidas se pueden utilizar para la autenticación”.

El AP puede especificar una palabra clave (acción) “Autenticar miembro” con un parámetro “tarjeta de miembro” y una palabra clave de verificación “Verificar acceso”. El AP comprueba si las palabras clave identificadas residen en la capa de abstracción adecuada.

Al especificar las palabras clave, el AP debe tener en cuenta que estas deben:

- contener un verbo (+ sustantivo).
- utilizar la forma imperativa del verbo (+ sustantivo).
- tener un significado único.
- estar adecuadamente documentadas.
- reflejar el vocabulario del dominio de la aplicación.
- ser reutilizables.

Al redactar los casos de prueba basados en palabras clave, el AP puede reconocer algunas palabras clave que faltan y especificarlas inmediatamente. Los casos de prueba basados en palabras clave pueden redactarse en diversos formatos, como listas o tablas.

Las palabras clave pueden cambiar a lo largo de la vida útil de un proyecto y son propensas a especificarse de forma redundante (Rwemalika et al., 2019). Las recomendaciones mencionadas en esta sección tienen como objetivo evitar la redundancia y reducir los esfuerzos de mantenimiento.

Aunque la prueba guiada por palabras clave es un enfoque de automatización de la prueba, la prueba manual también puede beneficiarse de ella. Si se aplica a la prueba manual, apoya eficazmente una transición posterior de la prueba manual a la automatizada.

Se puede encontrar más información sobre la automatización de la ejecución de la prueba y las pruebas guiadas por palabras clave en (ISTQB-TTA, v4.0), (ISTQB-Ingeniero de Automatización de Pruebas (IAP), v2.0) e (ISO/IEC/IEEE 29119-5, 2016).

1.3.7 Herramientas aplicadas en la gestión de productos de prueba

Una gestión adecuada de los productos de prueba mediante herramientas puede ayudar al AP a dar soporte al proceso de prueba en su conjunto. Las herramientas pueden proporcionar el estado de los productos de trabajo y dar soporte a la supervisión y el control de las pruebas. En caso de fallos en el sistema sometido a prueba, permiten al AP comprobar los resultados de pruebas anteriores y analizar el momento en que se han producido los defectos.

Algunos tipos clave de herramientas para la gestión de productos de prueba son:

- **Herramientas de gestión de la prueba** para proporcionar un repositorio de todos los productos de prueba relevantes (por ejemplo, condiciones de prueba, casos de prueba, guiones de prueba, conjuntos de prueba y ejecuciones de prueba). Las herramientas facilitan la trazabilidad (por ejemplo, a través de una matriz de trazabilidad), la recuperación de casos de prueba, el calendario de ejecuciones de prueba, el registro de resultados de prueba y la presentación de informes generales sobre el progreso y la calidad de las pruebas.
- **Herramientas de gestión de defectos** para registrar, priorizar y supervisar el proceso de resolución de defectos.
- **Herramientas de gestión de datos de prueba** para crear y mantener datos de prueba, incluida la protección de datos sensibles (véase la sección 1.3.5).
- Herramientas de gestión de la configuración para facilitar las actividades relacionadas con la prueba dentro de los procesos de desarrollo, entrega y operación, incluida la gestión de la configuración y la disponibilidad de los entornos de prueba.
- **Herramientas de gestión de requisitos** para recopilar y realizar un seguimiento de los requisitos de alto nivel, asegurando que estén claramente definidos, versionados y trazados a lo largo del ciclo de vida de desarrollo de software (CVDS) [software development lifecycle (SDLC)].

El AP apoya la gestión de productos de prueba al:

- Llevar a cabo un análisis del proyecto y la versión para seleccionar el subconjunto correcto de los productos de prueba (por ejemplo, una especificación identificada por su versión para que coincida con la versión del sistema sujeto a prueba (SSP)).
- definir una estructura funcional (por ejemplo, por prestaciones o módulos) o técnica (por ejemplo, por tipo de prueba o entorno) adecuada para la organización de los casos de prueba en la herramienta de gestión de pruebas.
- agregar metadatos a los casos de prueba (por ejemplo, información sobre el esfuerzo relacionado con la ejecución de la prueba o el entorno de prueba específico requerido).
- asegurar la trazabilidad entre requisitos, condiciones de prueba, pruebas, ejecuciones de pruebas y defectos.
- seleccionar los conjuntos de prueba correctos para las pruebas de regresión (para la ejecución de pruebas manuales/automatizadas).
- aplicar gestión de la configuración a los casos de prueba, incluida la identificación de casos de prueba desactualizados.

Las herramientas ayudan a organizar, realizar un seguimiento y garantizar la calidad del proceso de prueba. El AP apoya este esfuerzo seleccionando, estructurando y manteniendo los casos de prueba, garantizando la trazabilidad y gestionando las versiones de los casos de prueba para una prueba y un control eficientes.

2 Tareas del analista de prueba en la prueba basada en el riesgo

Duración: 90 minutos

Palabras Clave[2]

En la siguiente tabla se presentan las palabras clave del capítulo. En este documento, se identifican dos tipos de palabras clave:

- **ISTQB**: identificarán palabras clave del proceso de prueba
- **ESPDOM**: identificarán palabras clave específicas de dominio

Tipo Palabra Clave	Español	Inglés
ISTQB	análisis de impacto	impact analysis
ISTQB	riesgo de producto	product risk
ISTQB	prueba de regresión	regression testing
ISTQB	análisis del riesgo	risk analysis
ISTQB	evaluación del riesgo	risk assessment
ISTQB	control del riesgo	risk control
ISTQB	identificación de riesgos	risk identification
ISTQB	mitigación del riesgo	risk mitigation
ISTQB	monitorización del riesgo	risk monitoring
ISTQB	prueba basada en el riesgo	risk-based testing

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

Objetivos de Aprendizaje para "Capítulo 2"

2.1 Análisis del riesgo

TA-2.1.1 (K2) Resumir la contribución del analista de pruebas al análisis del riesgo de producto.

2.2 Control del riesgo

TA-2.2.1 (K4) Analizar el impacto de cambios para determinar el alcance de la prueba de regresión.

Introducción a las tareas del analista de pruebas en la prueba basada en el riesgo

La prueba basada en el riesgo es un enfoque de prueba que prioriza los esfuerzos de prueba en función de los niveles de riesgo de los elementos de prueba. El jefe de prueba determina este enfoque. El AP desempeña un rol activo en su implementación. La prueba basada en el riesgo se trata con más detalle en (ISTQB-TM, v3.0).

2.1 Análisis del riesgo

Según (ISTQB-CTFL, v4.0.1, Sección 5.2), el análisis del riesgo implica la identificación de riesgos y la evaluación del riesgo. Este programa de estudio se centra en cómo el AP debe participar en las actividades de análisis del riesgo para asegurar que las pruebas basadas en el riesgo se implementan correctamente.

2.1.1 La contribución del analista de prueba al análisis del riesgo de producto

El analista de prueba suele poseer conocimientos únicos sobre el sistema, así como experiencia y conocimientos intuitivos sobre lo que suele fallar, qué impacto tiene y cómo las pruebas pueden mitigar el riesgo. Esto convierte al analista de prueba en un implicado valioso para el análisis del riesgo de producto.

En la **identificación del riesgo**, el AP contribuye con su propia experiencia y conocimientos a las retrospectivas, los talleres sobre riesgos, la tormenta de ideas y la creación de listas de comprobación. El AP también puede realizar entrevistas con los implicados para comprender mejor cuáles son los riesgos que consideran más importantes desde su punto de vista.

Durante la evaluación del riesgo, el AP, junto con otros implicados, contribuye a la determinación del nivel de riesgo estimando varios factores como:

- frecuencia de uso y criticidad de las prestaciones afectadas.
- criticidad de los objetivos de negocio afectados.
- daños financieros, medioambientales y de reputación.
- calidad de la base de prueba.
- necesidades legales o de protección.

El AP también ayuda a clasificar los riesgos de producto en categorías según las características de calidad afectadas, por ejemplo, utilizando el modelo de calidad del producto de la norma ISO/IEC 25010 (2023). A menudo, el nivel de riesgo no está distribuido uniformemente en el objeto de prueba. En tales casos, el AP debe desglosar el objeto de prueba en elementos de prueba (por ejemplo, componentes, interfaces y prestaciones) y evaluar un riesgo determinado para cada elemento de prueba por separado.

Por último, como parte de la evaluación del riesgo de producto, el AP propone actividades de prueba adecuadas para mitigar cada riesgo de producto identificado. Estas actividades pueden incluir pruebas estáticas y pruebas dinámicas. En función de factores como el nivel de riesgo, el tipo de elemento de prueba asociado y la característica de calidad afectada, el AP indica los niveles de prueba necesarios, los tipos de prueba, las técnicas de prueba, los niveles de independencia de la prueba y la minuciosidad de la prueba. En el espíritu del desplazamiento a la izquierda, el AP indica qué actividades de prueba pueden mitigar el riesgo más temprano para minimizar el esfuerzo de prueba.

2.2 Control del riesgo

Según (ISTQB-CTFL, v4.0.1, Sección 5.2.4), el control del riesgo implica la mitigación del riesgo y la monitorización del riesgo. El AP comprende las funcionalidades del sistema y los riesgos potenciales relacionados con ellas. Por lo tanto, el rol del AP es crucial en varias acciones de mitigación del riesgo mencionadas en (ISTQB-CTFL, v4.0.1, Sección 5.2.4):

- La realización de revisiones se trata en la Sección 5.2.2 de este programa de estudio.
- La aplicación de las técnicas de prueba y los niveles de cobertura adecuados se trata en la Sección 3.5.
- La aplicación de los tipos de prueba adecuados se trata en el capítulo 4.
- La realización de pruebas de regresión se trata más adelante.

Además, como los riesgos y los niveles de riesgo no son estáticos y cambian con el tiempo, la prueba basada en el riesgo implica una monitorización del riesgo periódica. En un ciclo de vida iterativo, esto se realiza con una frecuencia determinada por el equipo (probablemente una vez por iteración). En otros ciclos de vida, su frecuencia la establece la persona responsable de la gestión del riesgo de producto, a menudo el jefe de pruebas. El AP contribuye actualizando el registro de riesgos en función de los cambios realizados y ajustando las acciones de mitigación del riesgo mencionadas anteriormente.

2.2.1 Determinación del alcance de la prueba de regresión

El objetivo principal de la prueba de regresión es asegurar la confianza en la calidad del objeto de prueba después de realizar un cambio. Sin embargo, puede resultar imposible ejecutar todas las pruebas de regresión durante la prueba de regresión debido a restricciones (por ejemplo, restricciones de tiempo, presupuesto, entorno de prueba o datos de prueba). Este es un asunto de interés principalmente en las pruebas de ejecución manual, pero también se aplica a las pruebas automatizadas, por ejemplo, cuando los ciclos de prueba son cortos y hay muchas pruebas automatizadas de larga duración. Esto hace necesario seleccionar pruebas de regresión adecuadas basadas en criterios específicos. Es esencial revisar el alcance de las pruebas de regresión con cada ciclo de prueba. La revisión puede concluir que es necesario ajustar el juego de pruebas de regresión existente.

La técnica más fiable para seleccionar las pruebas automatizadas es un análisis de impacto, que pueden soportar las herramientas. Estas herramientas se basan en un sistema automatizado de gestión de la configuración. Registran qué elementos de la configuración se activan durante la ejecución de cada caso de prueba. Cuando se produce un cambio, rastrean qué elementos de la configuración se han modificado y seleccionan pruebas de regresión que interactúen con los elementos modificados. De este modo, las herramientas aseguran que, tras un cambio en un elemento de la configuración, las pruebas se concentren en las áreas en las que es más probable encontrar fallos (Juergens et al., 2018).

En lo que respecta a la ejecución manual de pruebas, no ha habido evidencias concluyentes de una técnica de selección de pruebas de regresión que sea claramente superior, ya que los resultados dependen de varios factores (Engström et al., 2010). Por lo tanto, el AP debe decidir qué técnica utilizar en función de la situación dada. Entre las técnicas más utilizadas se incluyen:

- Selección de pruebas basada en el riesgo, en la que el AP mantiene la trazabilidad del juego de pruebas de regresión con un registro de riesgos. Cuando se realiza un cambio, y el registro de riesgos se actualiza en consecuencia, el AP ajusta el conjunto de pruebas de regresión para cubrir los niveles de riesgo más elevados.
- Pruebas basadas en históricos, en las que el AP evalúa las ejecuciones de pruebas anteriores y determina cuáles han expuesto defectos o han sido sensibles a cambios similares a los realizados

desde la última ejecución de prueba. Ejecutar de nuevo las pruebas correspondientes como parte de la prueba de regresión aumenta la probabilidad de exponer defectos similares. El AP también puede incluir algunas pruebas que no se hayan ejecutado durante mucho tiempo para asegurar que siguen pasando.

- Pruebas basadas en la cobertura, en las que el AP selecciona un pequeño número de pruebas que consigan la mayor cobertura posible en función de la(s) técnica(s) de prueba elegida(s). La cantidad de pruebas debe equilibrarse cuidadosamente con el aumento de cobertura por prueba.
- La matriz de trazabilidad de requisitos, que se utiliza para probar el impacto de los cambios de requisitos en las pruebas asociadas. Puede ser especialmente valiosa cuando los requisitos nuevos o modificados repercuten indirectamente en las prestaciones existentes. El AP selecciona pruebas de regresión para las directamente afectadas y para las prestaciones relacionadas, a fin de cubrir posibles efectos secundarios no deseados. En el desarrollo ágil de software, esto puede hacerse de forma similar seleccionando pruebas que cubran los criterios de aceptación afectados por las historias de usuario nuevas o modificadas.
- Pruebas basadas en perfiles operativos, en las que el AP selecciona los casos de prueba de regresión que deben ejecutarse en función de los patrones de uso del objeto de prueba. Por ejemplo, al probar una tienda en línea, una de esas pruebas puede incluir el registro del usuario, la búsqueda de productos, su adición al carrito y la realización del pedido. Cuando una aplicación sufre cambios importantes, esta técnica proporciona una visión rápida de la funcionalidad general del sistema. Si esto da lugar a demasiadas pruebas, el AP prioriza los patrones críticos de uso que se producen a menudo y cubren funcionalidades y procesos de negocio críticos.
- El análisis de impacto, que también puede utilizarse para seleccionar casos de prueba de regresión para su ejecución manual si el AP sabe cuáles de ellos interactúan con los elementos de configuración modificados.

A menudo es necesario utilizar una combinación de técnicas de selección para obtener un juego de pruebas de regresión más completo y eficaz. Sin embargo, el AP debe establecer un cuidadoso equilibrio entre la necesidad de cobertura y un tamaño manejable del juego de pruebas. Después de cada ciclo de prueba, el AP analiza los resultados de la prueba para determinar la efectividad de las técnicas aplicadas (véase la sección 5.3.1). A continuación, el AP conserva las técnicas eficaces y sustituye las ineficaces. De este modo se mejora continuamente la selección de pruebas de regresión a lo largo del tiempo. Es esencial en los modelos de desarrollo iterativo e incremental, en los que los cambios son frecuentes.

3 Análisis de prueba y diseño de prueba

Duración: 615 minutos

Palabras Clave[3]

En la siguiente tabla se presentan las palabras clave del capítulo. En este documento, se identifican dos tipos de palabras clave:

- **ISTQB**: identificarán palabras clave del proceso de prueba
- **ESPDOM**: identificarán palabras clave específicas de dominio

Tipo Palabra Clave	Español	Inglés
ISTQB	prueba basada en lista de comprobación	checklist-based testing
ISTQB	técnica de prueba basada en el comportamiento	behavior-based test technique
ISTQB	prueba combinatoria	combinatorial testing
ISTQB	prueba multitudinaria	crowd testing
ISTQB	prueba CRUD	CRUD testing
ISTQB	técnica de prueba basada en datos	data-based test technique
ISTQB	prueba de tabla de decisión	decision table testing
ISTQB	prueba de dominio	domain testing
ISTQB	partición de equivalencia	equivalence partition
ISTQB	prueba basada en la experiencia	experience-based testing
ISTQB	relación metamórfica	metamorphic relation

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

Tipo Palabra Clave	Español	Inglés
ISTQB	prueba metamórfica	metamorphic testing
ISTQB	prueba aleatoria	random testing
ISTQB	técnica de prueba basada en reglas	rule-based test technique
ISTQB	prueba basada en escenario	scenario-based testing
ISTQB	prueba basada en sesiones	session-based testing
ISTQB	prueba de transición de estado	state transition testing
ISTQB	contrato de prueba	test charter

Objetivos de Aprendizaje para "Capítulo 3"

3.1 Técnicas de prueba basadas en datos

TA-3.1.1 **(K3)** Aplicar la prueba de dominio.

TA-3.1.2 **(K3)** Aplicar la prueba combinatoria.

TA-3.1.3 **(K2)** Resumir las ventajas y limitaciones de la prueba aleatoria.

3.2 Técnicas de prueba basadas en el comportamiento

TA-3.2.1 **(K2)** Explicar la prueba CLAB (por acrónimo en inglés «CRUD»).

TA-3.2.2 **(K3)** Aplicar la prueba de transición de estado.

TA-3.2.3 **(K3)** Aplicar la prueba basada en escenarios.

3.3 Técnicas de prueba basadas en reglas

TA-3.3.1 **(K3)** Aplicar la prueba de tabla de decisión.

TA-3.3.2 **(K3)** Aplicar la prueba metamórfica.

3.4 Prueba basada en la experiencia

TA-3.4.1 **(K3)** Preparar contratos de prueba para la prueba basada en sesiones.

TA-3.4.2 **(K3)** Preparar listas de comprobación que apoyen la prueba basada en la experiencia.

TA-3.4.3 **(K2)** Aportar ejemplos de las ventajas y limitaciones de la prueba multitudinaria.

3.5 Aplicación de las técnicas de prueba más adecuadas

TA-3.5.1 **(K4)** Seleccionar las técnicas de prueba adecuadas para mitigar riesgos de producto en una situación determinada.

TA-3.5.2 **(K2)** Explicar las ventajas y los riesgos de automatizar el diseño de prueba.

Introducción al análisis de prueba y al diseño de prueba

Las técnicas de prueba se utilizan principalmente en el análisis de prueba y el diseño de prueba. Las técnicas de prueba que se tratan en este capítulo abarcan las técnicas de prueba de caja negra y las técnicas de prueba basadas en la experiencia. Las técnicas de prueba de caja blanca se tratan en (ISTQB-TTA, v4.0).

Este programa de estudio subdivide las técnicas de prueba de caja negra en tres categorías basadas en las condiciones de prueba subyacentes que modelan:

- elementos de los datos (basadas en datos).
- elementos del comportamiento dinámico (basadas en el comportamiento).
- elementos de las reglas de comportamiento estático (basadas en reglas).

3.1 Técnicas de prueba basadas en datos

En esta sección, el término «dominio» se utiliza para representar el conjunto de datos de entrada de un elemento de prueba. Los datos de entrada de diversas áreas del dominio conducen a diversos comportamientos del elemento de prueba. Las técnicas de prueba basadas en datos pretenden verificar que la implementación trata correctamente áreas específicas del dominio. El programa de estudio de nivel básico (ISTQB-CTFL, v4.0.1, Sección 4.2) cubre dos técnicas de prueba que pueden clasificarse como basadas en datos: partición de equivalencia (PE) y análisis del valor frontera (AVF). Este programa de estudio introduce otras tres técnicas de prueba basadas en datos:

- prueba de dominios - amplía el PE y el AVF a dominios con múltiples parámetros y particiones complejas.
- prueba combinatoria - se concentra en las interacciones de múltiples parámetros en un dominio multidimensional.
- prueba aleatoria - selecciona entradas aleatorias del dominio basándose en una distribución de probabilidad especificada.

Hay que tener en cuenta que la prueba aleatoria puede referirse en general tanto a datos de entrada aleatorios como a sucesos aleatorios. Este programa de estudio sólo se ocupa de las pruebas con datos de entrada aleatorios.

3.1.1 Prueba de dominio

La prueba de dominio verifica si el elemento de prueba se comporta como se especifica en las particiones de equivalencia del dominio y en sus fronteras. En este caso, las particiones de equivalencia se definen mediante expresiones que combinan condiciones atómicas mediante operadores booleanos (por ejemplo, AND, OR y NOT) e implican una o varias variables que interactúan. Cada condición atómica define una frontera de la partición de equivalencia. Las fronteras cerradas están formadas por expresiones relacionales con operadores ≤, ≥ o =, y las fronteras abiertas están formadas por expresiones relacionales con operadores <, >, o ≠.

Por ejemplo, la expresión **altura > 1,29 metros** AND **peso / altura^2 ≥ 30** define una partición de equivalencia de un dominio bidimensional de dos variables que tiene dos fronteras, una abierta y otra cerrada.

La prueba de dominio trata de identificar defectos en las implementaciones de particiones de equivalencia (por ejemplo, operador o constante erróneos) seleccionando elementos de cobertura adecuados que

puedan revelar esos defectos. Los criterios de cobertura en las pruebas de dominio se refieren a los puntos ON, OFF, IN y OUT:

- Para una frontera cerrada, un punto ON se encuentra en esta frontera. Para una frontera abierta, un punto ON se encuentra dentro de la partición de equivalencia y está más cerca de la frontera según la precisión dada.
- Para una frontera cerrada, un punto OFF se encuentra fuera de la partición de equivalencia y está más cerca de la frontera según la precisión dada. Para una frontera abierta, un punto OFF se encuentra en esta frontera.
- Un punto IN pertenece a la partición de equivalencia y no es un punto ON.
- Un punto OUT se encuentra fuera de la partición de equivalencia y no es un punto OFF.

Cada uno de estos tipos de puntos está relacionado con la frontera de la partición de equivalencia. Un mismo punto puede tener diferentes tipos para diferentes bordes.

- Los criterios de cobertura incluyen:
  - **Cobertura de dominio simplificada** (Jeng et al., 1994), que requiere los siguientes elementos de cobertura:
    - Un punto ON y un punto OFF para cada frontera definida por uno de los operadores <, ≤, > o ≥.
    - Un punto ON y dos puntos OFF situados a distintos lados de la frontera para el = operador.
    - Un punto OFF y dos puntos ON situados a diferentes lados de la frontera para el operador ≠.

    Cada punto OFF deberá estar lo más cerca posible del punto ON correspondiente según la precisión dada.
  - **Cobertura de dominio fiable** (Forgács et al., 2024), que requiere los siguientes elementos de cobertura:
    - Un punto ON, un punto OFF, un punto IN y un punto OUT para cada frontera definida por uno de los operadores <, ≤, > o ≥.
    - Un punto ON y dos puntos OFF situados a distintos lados de la frontera para el = operador.
    - Un punto OFF y dos puntos ON situados a distintos lados de la frontera para el operador ≠.

Para los dos criterios de cobertura comentados, el número de elementos de cobertura depende linealmente del número de fronteras. Se puede optimizar para ambos criterios de cobertura utilizando el mismo elemento de cobertura para diferentes bordes. Por ejemplo, todas las fronteras de una partición de equivalencia pueden utilizar un punto IN común o un par de puntos ON y OFF de una frontera pueden utilizarse como puntos OFF y ON para la frontera de la partición de equivalencia adyacente. Para dominios con muchas fronteras, se dispone de un algoritmo de optimización avanzado (Forgács et al., 2024).

La cobertura de dominio fiable da como resultado una cantidad ligeramente mayor de elementos de cobertura que la cobertura de dominio simplificada, pero puede detectar considerablemente más defectos de dominio (Site of Software Test Design, 2020).

La prueba de dominio puede aplicarse a cualquier nivel de prueba. Generaliza AVF y PE (véase ISTQB-CTFL, v4.0.1, Sección 4.2) a dominios más complejos. En libros de texto como (Beizer, 1990, Cap. 6; Binder, 2000, Sección 10.2.4; Kaner; Padmanabhan, et al., 2013; Jorgensen, 2014, Cap. 5) se pueden encontrar varios enfoques para probar dominios.

3.1.2 Prueba combinatoria

Algunos fallos de software surgen de combinaciones específicas de valores de parámetros, lo que se conoce como fallos de interacción. La prueba combinatoria pretende revelar estos fallos explorando estas combinaciones de valores de parámetros.

En la prueba combinatoria, las condiciones de prueba suelen combinar parámetros de configuración o valores de datos de entrada. Por lo tanto, existen dos enfoques principales para la prueba combinatoria. El primer enfoque utiliza combinaciones de parámetros de configuración, y cada una de estas combinaciones puede probarse utilizando el mismo caso de prueba. El segundo enfoque utiliza combinaciones de valores de datos de entrada, que luego pasan a formar parte de casos de prueba completos, creando un juego de pruebas para el sistema sujeto a prueba (D. R. Kuhn et al., 2013).

Un par parámetro-valor específico consiste en un parámetro y su valor (por ejemplo, '(color, rojo)').

Entre los criterios de cobertura combinatoria se incluyen los siguientes (Ammann et al., 2008), (Forgács et al., 2019):

- La **cobertura de opción base** asume que algunos pares parámetro-valor son más importantes que otros. Se elige un par parámetro-valor base para cada parámetro, y un elemento de cobertura base es la combinación de pares parámetro-valor base. Los elementos de cobertura posteriores se crean a partir del elemento de cobertura base para cada parámetro sustituyendo su par parámetro-valor base por cada valor no base.
- **Cobertura por parejas**, en la que los elementos de cobertura son pares parámetro-valor para dos parámetros cualesquiera. Existen herramientas para generar elementos de cobertura. Sin embargo, encontrar un conjunto mínimo de casos de prueba que consigan una cobertura por pares suele ser difícil.

En el caso de parámetros con muchos valores, se puede aplicar primero la PE para reducir este número y el conjunto de combinaciones resultantes. La captura de los parámetros y sus valores en un árbol de clasificación o en un modelo de prestación (véase IREB-Glossary, 2024) apoya esta actividad. El número final de casos de prueba puede verse afectado por restricciones entre pares parámetro-valor, por combinaciones añadidas manualmente que se sabe que son problemáticas, o debido a combinaciones no válidas o inviables.

La idea crítica para las pruebas combinatorias es que no todos los parámetros contribuyen a un fallo y que la mayoría de los fallos se desencadenan por un único valor de parámetro o por interacciones entre un número relativamente pequeño de parámetros (Cohen et al., 1994). Esto se alinea con la hipótesis del efecto de acoplamiento, que indica que la detección de defectos simples en un programa a menudo puede descubrir también defectos complejos (Offutt, 1992). En un estudio limitado (D. Kuhn et al., 2004), los resultados mostraron que alrededor del 97% de los fallos están causados por sólo una o dos condiciones que interactúan, lo que indica que la prueba de a pares es una técnica de prueba eficaz.

Puede encontrar más información sobre prueba combinatoria, incluidos otros criterios de cobertura como **diff-pair-t**[4] o **n-wise testing**[5], en (Ammann et al., 2008), (Forgács et al., 2019). También hay disponibles herramientas de apoyo a las pruebas combinatorias en (Czerwonka, 2004).

3.1.3 Prueba aleatoria

La prueba aleatoria consiste en seleccionar datos de prueba al azar del dominio de entrada del elemento de prueba basándose en una distribución de probabilidad especificada. A efectos de validación, se recomienda una distribución basada en perfiles operativos. Para fines de verificación, la distribución debe ser agnóstica respecto al uso para evitar sesgos. Los resultados esperados pueden añadirse a los casos de prueba mediante un oráculo de prueba, que la mayoría de las veces requiere un oráculo de prueba automatizado.

La prueba aleatoria puede ser guiada o no guiada. En la prueba aleatoria no guiada, la distribución de probabilidades permanece fija durante todo el proceso. La prueba aleatoria guiada, de la que son ejemplo técnicas como la prueba aleatoria adaptativa (Huang et al., 2019), ajusta la distribución basándose en valores previamente seleccionados, evolucionando con el tiempo. La prueba aleatoria guiada pretende cubrir el dominio de entrada de forma efectiva, teniendo en cuenta que los defectos suelen agruparse en regiones específicas del dominio.

La prueba aleatoria carece de criterios de cobertura reconocidos. Por lo tanto, los criterios de salida sólo pueden basarse en el número de pruebas ejecutadas, el tiempo de prueba o medidas similares de compleción.

La prueba aleatoria es especialmente valiosa cuando el conocimiento del dominio es limitado o se necesita un gran volumen de datos de prueba. Es rentable y proporciona, en términos probabilísticos, información sobre la fiabilidad del objeto de prueba. La prueba aleatoria ayuda a evitar sesgos como pasar por alto defectos en las pruebas manuales debido a una confianza equivocada en algún código o funcionalidad. Sin embargo, la prueba aleatoria también tiene varios retos y limitaciones, como descuidar la semántica de los datos, omitir potencialmente defectos relacionados con el significado de los datos, pasar por alto ciertos defectos, generar pruebas redundantes, depender de un oráculo de prueba automatizado y salidas aleatorias que conducen a resultados de prueba incoherentes. Equilibrar las ventajas y las limitaciones de las pruebas aleatorias es esencial en cada contexto de prueba.

3.2 Técnicas de prueba basadas en el comportamiento

Las técnicas de prueba basadas en el comportamiento obtienen casos de prueba a partir de especificaciones del comportamiento dinámico (es decir, dependiente del estado) del elemento de prueba. En este programa de estudio se analizan tres técnicas de prueba basadas en el comportamiento:

- prueba CLAB (por acrónimo en inglés «CRUD»).
- prueba de transición de estado.
- prueba basada en escenarios.

[4] Ver anexo: Traducciones específicas, definiciones, acrónimos, abreviaturas y notas

[5] Ver anexo: Traducciones específicas, definiciones, acrónimos, abreviaturas y notas

La prueba CLAB (por acrónimo en inglés «CRUD») y la prueba basada en escenarios amplían la gama de técnicas de prueba de caja negra conocidas por (ISTQB-CTFL, v4.0.1, Sección 4.2). Este programa de estudio complementa la prueba de transición de estado con criterios de cobertura adicionales.

3.2.1 Prueba CLAB

La prueba CLAB (por acrónimo en inglés «CRUD») verifica el ciclo de vida de las entidades de datos procesadas por el elemento de prueba. CLAB significa crear, leer, actualizar y borrar. Estas son las cuatro operaciones básicas que las funciones pueden realizar sobre las entidades.

La matriz CLAB ofrece una visión general del ciclo de vida de las entidades de datos. Sus columnas representan las entidades y sus filas las funciones. Se puede suponer que una función ejecuta una o varias operaciones concretas de creación, lectura, actualización o borrado sobre una entidad determinada. Esto se muestra en la matriz utilizando las iniciales de estas operaciones: C, L, A o B. Para crear una matriz CLAB, el AP determina para cada función cuál de las cuatro operaciones realiza sobre qué entidades. Debe prestarse especial atención a la operación de lectura, a menudo vinculada implícitamente a las operaciones C-A-B.

La prueba CLAB (por acrónimo en inglés «CRUD») consta de dos partes (Koomen et al., 2006):

- **prueba de completitud**
    - La **prueba de completitud** es una prueba estática que verifica si todas las operaciones posibles (es decir, C, R, U y D) se producen con cada entidad (es decir, si se ha implementado todo el ciclo de vida para cada entidad). La ausencia de una operación es una anomalía que requiere investigación.
- **prueba de consistencia**
    - La **prueba de consistencia** es una prueba dinámica cuyo objetivo es integrar las distintas funciones y comprobar si la entidad se utiliza de forma consistente. Verifica que las funciones interactúan correctamente al tratar una entidad. Los casos de prueba deben cubrir todas las operaciones de la matriz CLAB. Además, también deben incluirse pruebas negativas (por ejemplo, la lectura de una entidad que aún no se ha creado). Los casos de prueba se diseñan por entidad combinando funciones para cubrir todo su ciclo de vida.

La cobertura CLAB se mide por el número de operaciones ejecutadas por el juego de prueba dividido por el número total de operaciones de la matriz CLAB. Una cobertura CLAB más rigurosa puede tener en cuenta combinaciones específicas de operaciones como elementos de cobertura (por ejemplo, después de cada A, deben cubrirse todas las L posibles).

3.2.2 Prueba de transición de estado

Muchos sistemas complejos están llenos de estados (es decir, la reacción del sistema ante un evento depende del estado actual del sistema). Ejemplos de sistemas con estado son los sistemas embebidos, los sistemas basados en diálogo, los sistemas de control o los sistemas que tratan con entidades y sus ciclos de vida.

Un estado simplifica los detalles internos complejos, lo que facilita a los implicados la comprensión del comportamiento esperado. Durante el análisis de prueba, el AP debe asegurar que el modelo basado en estados representa el comportamiento esperado del elemento de prueba al nivel de detalle necesario para probarlo. Si la base de prueba ya contiene un modelo de este tipo, el AP debe comprobar si contiene las condiciones de prueba y, si es necesario, adaptarlo o diseñar un nuevo modelo.

En los modelos basados en estados, los nodos representan estados y las aristas, transiciones de estado. Las transiciones de estado se desencadenan por eventos y pueden incluir condiciones de guarda y

acciones (véase ISTQB-CTFL, v4.0.1, sección 4.2.4). Se utilizan diversas variantes de estos modelos, como las máquinas de estados finitos ampliadas (Bochmann et al., 1994), los gráficos de estados de Harel (Harel, 1987) o las máquinas de estados UML (OMG® UML, 2017).

Además de los criterios de cobertura de transición de estado que se tratan en (ISTQB-CTFL, v4.0.1), a continuación, se tratan otros dos criterios para los que se ha demostrado empíricamente una alta efectividad de detección de defectos:

La cobertura de conmutador de orden N se aplica a secuencias válidas de N+1 transiciones consecutivas, también denominadas N-conmutaciones («switch») (Chow, 1978). La cobertura de 0-conmutaciones es igual a la cobertura de transiciones válidas (véase ISTQB-CTFL, v4.0.1, Sección 4.2.4). Una 1-conmutación es un par de transiciones entrantes y salientes en un estado. La ampliación de N-conmutadores en su extremo mediante transiciones de estado posteriores válidas da como resultado N+1 conmutadores. Las coberturas de 0-conmutación y 1-conmutación se utilizan con frecuencia en la práctica. Una cobertura del 100% de 2-conmutaciones o superior sólo está indicada para un alto riesgo de fallo debido a secuencias de eventos inesperados, ya que el número de N conmutadores puede crecer exponencialmente con N.

La cobertura de trayectos circulares (Ammann et al., 2008) se aplica a los caminos de un modelo basado en estados que forman bucles. Un trayecto circular es un bucle en el que los estados inicial y final son el mismo, y ningún otro estado de este bucle ocurre dos veces. Los elementos de cobertura son los trayectos circulares. (Antoniol et al., 2002) encontraron este criterio muy eficaz para detectar defectos.

La prueba de transición de estado es muy adecuada para la automatización mediante herramientas de prueba basadas en modelos. Al igual que la mayoría de las técnicas de prueba de caja negra, la prueba de transición de estado contribuye a la prevención de defectos mediante la detección de defectos en la especificación durante el modelado. Las pruebas de transición de estado pueden aplicarse a cualquier nivel de prueba.

En (Rechtberger et al., 2022) se analizan otros criterios de cobertura de la transición de estado. Un modelo reciente basado en estados es el modelo acción-estado que se discute en (Forgács et al., 2024).

3.2.3 Prueba basada en escenarios

La prueba basada en escenarios evalúa el comportamiento del elemento de prueba en escenarios realistas. Técnicas como la investigación de usuarios, las historias de usuario, las personas y los mapas de recorrido de usuario (véase la sección 11) pueden ayudar a identificar escenarios valiosos. En las pruebas basadas en escenarios, el AP crea un modelo de escenario basado en secuencias de acciones que constituyen flujos de trabajo a través del elemento de prueba (véase ISO/IEC/IEEE 29119-4, 2021). En este programa de estudio se analizan los dos modelos siguientes: diagramas de actividad y casos de uso. Otros modelos incluyen diagramas de flujo, diagramas de modelo de proceso de negocio y notación (OMG® BPMN, 2013), diagramas de secuencia o diagramas de colaboración (OMG® UML, 2017). Estos modelos no se tratan en este programa de estudio. Por favor, consulte (ISTQB- AcT, v1.0) para más detalles.

Un diagrama de actividades es una representación gráfica del flujo de trabajo dentro de un sistema. Los diagramas de actividad son especialmente útiles para modelar procesos de negocio, pero también pueden modelar el flujo de control. Los diagramas de actividad amplían la notación de los gráficos de flujo, permitiendo modelar la concurrencia. Los elementos principales de los diagramas de actividad son los nodos de inicio y fin, las acciones, las transiciones, los nodos de decisión, los nodos de fusión, los nodos de bifurcación, los nodos de unión y los carriles de natación.

Un caso de uso es una descripción textual o gráfica de acciones que representan las interacciones entre un usuario y un sistema o entre sistemas. En el modelo se distinguen tres tipos de escenarios:

- **Escenario principal (denominado «camino feliz»):** una secuencia típica y esperada de acciones que conducen a la consecución de un objetivo específico desde la perspectiva del usuario. Un caso de uso sólo puede tener un escenario principal.
- **Extensión (o escenario alternativo):** una secuencia de acciones, distinta del escenario principal, que conduce finalmente a la consecución del objetivo del escenario principal.
- **Excepción:** una secuencia de acciones que no permite alcanzar el objetivo del escenario principal debido a una acción inesperada (por ejemplo, un uso anormal o una entrada no válida).

En la prueba basada en escenarios, el AP diseña casos de prueba para cubrir los escenarios (es decir, elementos de cobertura), a menudo siguiendo una priorización basada en el riesgo. Cuando el modelo de escenarios no contiene bucles, cada escenario puede probarse con un caso de prueba independiente (es decir, todos los escenarios posibles del modelo pueden practicarse con un conjunto de pruebas). El número de escenarios (caminos) potenciales puede ser infinito cuando existen bucles. En este caso, la cobertura de bucle simple puede aplicarse al modelo de escenarios. La cobertura de bucle simple requiere probar cuando cada bucle se ejecuta cero veces (es decir, se salta), con exactamente una iteración, con más de una iteración (es decir, un número típico de iteraciones) y con el número máximo de iteraciones (si es posible).

La cobertura basada en escenarios se mide por el número de escenarios ejecutados dividido por el número de todos los escenarios identificados. Los escenarios pueden identificarse aplicando diversos criterios de cobertura al modelo de escenarios (véase Koomen et al., 2006). Los criterios de cobertura pueden requerir probar cada escenario más de una vez. Por ejemplo, un escenario puede requerir una cobertura adicional de PE o AVF de las variables que se dan en el escenario. En tales casos, es posible que un escenario necesite ser probado por más de un caso de prueba.

La prueba basada en escenarios suele realizarse en la prueba de sistema o en la prueba de aceptación. Adoptan la forma de pruebas de extremo a extremo concentradas en la adecuación funcional de un sistema (véase la sección 4.1) desde la perspectiva del usuario. Sin embargo, la prueba basada en escenarios también puede utilizarse en otros niveles de prueba (por ejemplo, la prueba de integración de componentes con escenarios basados en los protocolos de interacción o la prueba de componentes de clases orientadas a objetos y con estado con escenarios que invocan diversos métodos) y en pruebas no funcionales (por ejemplo, los escenarios pueden constituir los elementos de los perfiles operativos utilizados en las pruebas de fiabilidad, flexibilidad o compatibilidad).

3.3 Técnicas de prueba basadas en reglas

Las técnicas de prueba basadas en reglas verifican la implementación del comportamiento sin estado de un elemento de prueba, especificado por reglas que son válidas independientemente de su estado (por ejemplo, las reglas de negocio). En este programa de estudio se analizan dos técnicas de prueba basadas en reglas, a saber:

- prueba de tabla de decisión.
- pruebas metamórficas.

El programa de estudio de nivel básico (ISTQB-CTFL, v4.0.1) cubre los fundamentos de la prueba de tabla de decisión. En este programa de estudio se tratan temas más avanzados. La terminología y la notación utilizadas siguen el estándar (OMG® DMN, 2024).

3.3.1 Prueba de tabla de decisión

En la prueba de tabla de decisión, el AP suele empezar creando una tabla de decisión completa o analizando una existente obtenida de la base de prueba. El número de reglas de una tabla de decisión completa es el producto del número de valores de sus condiciones. Este número crece exponencialmente con el número de condiciones y sus valores y motiva la minimización de las tablas de decisión.

Las tablas de decisión pueden minimizarse fusionando reglas utilizando el valor «indiferente»[6], '-'. Se recomienda no tener en cuenta las reglas inviables (es decir, reglas con una combinación de valores de condiciones que nunca puede darse) al fusionar reglas. A menudo, las reglas no viables se eliminan de la tabla de decisión antes de fusionarlas. Un posible algoritmo de minimización sistemática busca reglas equivalentes en acción que sólo difieran en una condición y abarquen todos sus valores posibles. Estas reglas se fusionan y el valor de la condición que difiere se sustituye por «-». El resultado de la minimización sistemática puede depender del orden en que se minimicen las columnas, por lo que no siempre conduce a una solución óptima. El AP tiene que comprobar si es posible otra minimización.

Es tarea del AP revisar la tabla de decisión, posiblemente con otros implicados, con los siguientes criterios:

- consistencia (es decir, si dos reglas diferentes se aplican a la misma combinación de valores de condición, entonces son equivalentes en acción)
- viabilidad (es decir, no contiene reglas inviables)
- completitud (es decir, no falta ninguna combinación factible de valores de condición)
- corrección (es decir, las reglas modelan el comportamiento previsto del sistema)

Adicionalmente, es aconsejable que las reglas no se solapen (es decir, para cualquier combinación de valores de condición, se aplica como máximo una regla). El solapamiento de reglas puede ocurrir cuando la tabla de decisión original ya está minimizada o si las reglas se fusionan de forma incorrecta.

El procedimiento de suma de comprobación utiliza el número de reglas de una tabla de decisión minimizada para indicar solapamientos y vacíos. Para cada regla de la tabla minimizada, se calcula el número de reglas que representa en la tabla de decisión original. Una regla sin un '-' en las condiciones (es decir, con valores individuales para todas las condiciones) puntúa 1. Cada valor '-' multiplica la puntuación de la regla por el número de valores individuales para la condición respectiva. La suma de las puntuaciones de las reglas es la suma de comprobación de la tabla de decisión minimizada. Si la suma de comprobación es menor que la suma de comprobación de la tabla de decisión original, la tabla de decisión minimizada está incompleta. Si la suma de comprobación es mayor, algunas reglas se solapan o existen reglas adicionales (por ejemplo, combinaciones inviables de condiciones). Si la minimización se ha realizado correctamente, las sumas de comprobación son iguales. Las sumas de comprobación iguales por sí solas no garantizan que la tabla de decisión minimizada sea equivalente a la original.

La cobertura de la tabla de decisión se mide dividiendo el número de columnas practicadas por el número total de columnas factibles de la tabla de decisión. Cuando se implementan casos de prueba resultantes de una regla de tabla de decisión, el AP debe implementar las condiciones y las acciones. El AP tiene que decidir cómo implementar los valores de condición «-» para una regla determinada porque «-» representa al menos dos valores. Sin embargo, si el nivel de riesgo asociado a la tabla de decisión es alto, el AP debe abstenerse de minimizar y debe medir la cobertura de la tabla de decisión de las columnas factibles en la tabla de decisión completa.

[6] Ver anexo: Traducciones específicas, definiciones, acrónimos, abreviaturas y notas

3.3.2 Prueba metamórfica

La prueba metamórfica (PM) es una técnica destinada a generar casos de prueba que se basan en un caso de prueba de origen existente. Uno o más casos de prueba de seguimiento se generan cambiando (metamorfoseando) el caso de prueba de origen basándose en una relación metamórfica (RM). La RM define una propiedad del elemento de prueba y describe cómo un cambio en las entradas de un caso de prueba se refleja en los resultados esperados del caso de prueba.

El AP combina los casos de prueba de origen y de seguimiento de una RM en un procedimiento de prueba, con una evaluación conjunta de los resultados. Si cumplen la relación metamórfica, la prueba pasa, de lo contrario falla. En caso de fallo, la depuración posterior debe determinar cuál de los casos de prueba individuales involucrados ha fallado.

Por ejemplo, tener en cuenta una función que determina la media de una serie de números. Se crea un caso de prueba fuente con una serie de números y una media esperada, y se ejecuta el caso de prueba para confirmar que pasa. Una RM podría establecer que cualquier permutación de la serie de números da como resultado la misma media.

Utilizando esta RM, el AP puede crear varios casos de prueba de seguimiento, cada uno con el mismo conjunto de números en la entrada pero en un orden diferente. El resultado esperado sigue siendo el mismo.

Para la misma función de promedio, el AP también puede utilizar otra RM, que establece que si cada número de la serie se multiplica por el mismo número, x, entonces el resultado esperado también se multiplicará por x. Con esta RM, el AP puede crear cualquier número de casos de prueba de seguimiento a partir de un caso de prueba de origen eligiendo diferentes valores de x. Esto puede ser especialmente útil cuando el diseño de la prueba y la ejecución de la prueba están automatizados. El AP también puede combinar dos o más RM para crear casos de prueba de seguimiento (por ejemplo, permutar y multiplicar por 2).

La PM también puede utilizarse en presencia de un problema del oráculo de prueba (véase la sección 1.3.4). En tal situación, los resultados esperados del caso de prueba de origen y de los casos de prueba de seguimiento no están disponibles, por lo que sus resultados de prueba no pueden evaluarse individualmente. Un ejemplo puede ser un programa actuarial basado en IA que predice la edad al morir basándose en un gran conjunto de datos. La RM podría establecer que si se aumenta el número de cigarrillos fumados, la edad prevista de fallecimiento debería disminuir.

En la actualidad, no existen medidas de cobertura reconocidas para la MT que proporcionen criterios de salida útiles. Cubrir cada RM una vez es insuficiente porque sólo se puede obtener una verificación parcial de los resultados esperados. El TA puede combinar la PM con una prueba aleatoria para generar muchos casos de prueba fuente de bajo nivel y casos de prueba de seguimiento para la misma RM. Por ejemplo, en el cálculo de la media mencionado anteriormente, se puede utilizar un generador de números aleatorios para generar varias entradas para el caso de prueba fuente, así como permutaciones y multiplicadores aleatorios para los casos de prueba de seguimiento.

La PM puede utilizarse para la mayoría de los elementos de prueba y aplicarse a pruebas funcionales y no funcionales (por ejemplo, pruebas de carga, generación de carga mediante casos de prueba de seguimiento utilizando RM, o pruebas de instalabilidad con diversos parámetros de instalación que pueden seleccionarse en secuencias múltiples). Es una técnica de prueba preferida para los sistemas basados en IA (ISTQB-AI, v1.0).

Para más detalles, consulte también el estándar (ISO/IEC/IEEE 29119-4, 2021) o los artículos de estudio (Segura; Towey, et al., 2020) y (Segura; Fraser, et al., 2016).

3.4 Prueba basada en la experiencia

Un AP emplea enfoques de prueba basados en la experiencia y técnicas de prueba, aprovechando su experiencia y encuentros pasados para guiar la prueba. En este capítulo se detalla el uso que hace el AP de la documentación en las pruebas basadas en sesiones y en listas de comprobación (véase ISTQB-CTFL, v4.0.1, secciones 4.4.2 y 4.4.3). Además, se trata la prueba multitudinaria. Todas las técnicas de prueba basadas en la experiencia pueden aplicarse en la prueba basada en la colaboración, como el desarrollo guiado por prueba de aceptación. Para más detalles, consulte (ISTQB-CTFL, v4.0.1, Sección 4.5).

3.4.1 Contratos de prueba que apoyan la prueba basada en sesión

En la prueba exploratoria, un contrato de prueba proporciona la misión de una sesión de prueba, que describe su alcance y objetivos e información como limitaciones, plazos y riesgos. Sirve como hoja de ruta para probar, proporcionando estructura a las sesiones de prueba. Los contratos de pruebas ayudan al AP a concentrarse en las áreas o prestaciones específicas que deben probarse, al tiempo que le permiten la flexibilidad de explorar el sistema cuando sea necesario. El contrato de prueba no especifica los conjuntos de pruebas que se ejecutarán en cada sesión de prueba.

Al preparar un contrato de prueba para una prueba basada en la sesión, el AP debe tener en cuenta ciertos factores que influyen en el diseño del contrato de prueba, en particular:

- factores relacionados con el cliente y los requisitos (por ejemplo, los requisitos educidos de los clientes, los casos de uso de negocio para el sistema, los requisitos de calidad) y los mapas del recorrido del usuario (es decir, la interacción del usuario con el sistema a lo largo del tiempo).
- factores del producto (por ejemplo, flujos funcionales, objetivos principales del producto, prestaciones del producto, diseño del software e interfaces).
- factores de gestión del proyecto (por ejemplo, restricciones de tiempo, propósito del proyecto, esfuerzo estimado y valor de negocio).

Un contrato de prueba consiste en una misión que describe el objetivo de prueba y diversa información adicional.

Un formato ligero popular de una misión es «**Explorar** [objetivo] **Con** [recursos] **Para descubrir** [información]» (Hendrickson, 2013), donde [**objetivo**] describe lo que se va a explorar (por ejemplo, área, prestación, riesgo, componente y requisito), [**recursos**] describe lo que utilizará el AP (por ejemplo, datos de prueba, configuraciones, herramientas, restricciones, heurística y dependencias), e [**información**] explica qué tipo de información pretende encontrar el AP (por ejemplo, evaluaciones de características de calidad, tipo de defectos esperados y violaciones de los estándares).

Los contratos de prueba pueden contener, aunque no exclusivamente, la siguiente información adicional (Ghazi et al., 2017):

- información organizativa (por ejemplo, duración de la sesión de prueba, fecha y hora de inicio y nombre del probador).
- objetivos de la prueba (por ejemplo, motivación de la prueba y misión del contrato de prueba).
- alcance de la prueba (por ejemplo, áreas específicas de interés dentro del sistema sometido a prueba, nivel de prueba, técnicas de prueba que se utilizarán, ideas de prueba, criterios de salida, prioridades, lo que se supone que cubre el contrato de prueba y una descripción de lo que no se probará).

- criterios de entrada (es decir, precondiciones que deben cumplirse para poder iniciar la sesión de prueba).
- información relacionada con el producto (por ejemplo, definición, datos y flujo de trabajo entre componentes, y arquitectura del sistema).
- limitaciones (es decir, lo que el producto no debe hacer nunca).
- descripción del entorno de prueba.
- fuentes de datos existentes, información sobre el producto y herramientas de prueba que ayudarían a probarlo.
- información histórica (por ejemplo, defectos encontrados anteriormente, como defectos de compatibilidad e interoperabilidad, preguntas abiertas actuales que hagan referencia a anomalías existentes y patrones de fallos relacionados con las pruebas en el pasado).
- restricciones y riesgos (por ejemplo, reglamentos, normas y estándares utilizados).

Durante una sesión de prueba, el AP lleva un registro de la prueba que incluye preguntas, observaciones o ideas para futuras pruebas junto con los resultados de la prueba. Estos datos se documentan en una hoja de sesión.

El alcance y el detalle de la información incluida en los contratos de prueba específicos pueden variar y afectar al grado de flexibilidad del AP. Por ejemplo, definir sólo los objetivos generales de la prueba ofrece un amplio margen de exploración, mientras que añadir la información sobre las técnicas de prueba que se utilizarán puede limitar al AP. Por otro lado, añadir información (por ejemplo, lo que el sistema definitivamente no puede hacer) puede ayudar a reducir la probabilidad de informar sobre resultados de falso positivo.

3.4.2 Listas de comprobación que apoyan las técnicas de prueba basadas en la experiencia

La prueba basada en listas de comprobación es una técnica de prueba muy utilizada debido a su adaptabilidad, sencillez y efectividad para asegurar la calidad del software. Mediante el uso de listas de comprobación, el AP asegura que cubre todos los aspectos esenciales conocidos de un elemento de prueba, lo que evita que se pasen por alto áreas críticas. También introducen consistencia a lo largo de los ciclos de prueba y entre varios AP. Al utilizar una lista de comprobación, el AP se concentra en los aspectos importantes. Las listas de comprobación son una forma de registrar las experiencias pasadas del AP con fallos y defectos. Sirven de recordatorio o de fuente de inspiración si el AP se queda sin ideas (por ejemplo, durante una prueba exploratoria). El AP ahorra tiempo reutilizando una lista de comprobación estándar o una lista de comprobación creada por uno de sus pares, pero sólo si estas listas de comprobación son relevantes para el elemento de prueba. Las listas de comprobación ayudan a reducir el trabajo necesario para documentar los casos de prueba, lo que puede ser un gran activo cuando los requisitos y el software cambian constantemente.

Con frecuencia, la preparación de las listas de comprobación es responsabilidad del AP. Las listas de comprobación apoyan las pruebas basadas en la experiencia, ya que ayudan a organizar, estructurar y guiar la prueba. Crear una lista de comprobación reutilizable, mantenible, clara y eficiente requiere esfuerzo.

Los siguientes pasos pueden servir de guía para crear una lista de comprobación adecuada para la prueba basada en la experiencia:

- En primer lugar, el AP determina el alcance, los objetivos y el formato de la lista de comprobación porque influyen en la profundidad de la prueba y en el nivel de detalle necesario. Una lista de

comprobación de leer-hacer contiene los principales elementos a tener en cuenta para un determinado proceso (por ejemplo, los elementos de la lista de comprobación contienen entradas no válidas concretas para un campo de entrada que hay que comprobar). Una lista de comprobación de hacer-confirmar sirve de ayuda para guiar el proceso de razonamiento, proporcionando ideas sobre la prueba basada en la experiencia para explorar más a fondo una aplicación (por ejemplo, un elemento de la lista de comprobación podría requerir comprobar si los resultados de la búsqueda son relevantes).

- A continuación, el AP recopila la información necesaria para definir los elementos de la lista de comprobación. Esto puede incluir la recopilación de información de profesionales con experiencia, la búsqueda en librerías de defectos y taxonomías de defectos (Beizer, 1990), (Kaner; Falk, et al., 1999), la revisión de la documentación pertinente y el análisis de riesgos, casos de prueba y escenarios potenciales. Los elementos de la lista de comprobación deben ser claros, específicos, inequívocos, consistentes, relevantes, mantenibles, procesables y medibles. Deben formularse como preguntas que puedan responderse con «sí», «no» o «no aplicable». Requieren que se les asignen prioridades en función de su importancia, impacto potencial y nivel de riesgo. Las listas de comprobación no son guías prácticas exhaustivas. Por el contrario, son herramientas rápidas y sencillas para profesionales expertos.
- Por último, el AP estructura y organiza la lista de comprobación clasificando los elementos de la lista en grupos lógicos basados en áreas funcionales, roles de usuario, niveles de prueba u otros criterios relevantes. La creación de categorías separadas puede ser especialmente útil para una lista de comprobación extensa.

Siempre que sea posible, deben utilizarse plantillas y estándares. El AP puede ahorrar esfuerzos utilizando plantillas existentes o listas de comprobación predefinidas que se ajusten a los estándares y las buenas prácticas del sector.

Una lista de comprobación nunca es definitiva. El AP la revisa y la perfecciona continuamente, adaptándola para reflejar nuevos hallazgos, cambios en las prioridades, retroalimentación de otros probadores o lecciones aprendidas en ciclos de prueba anteriores. Al compartir la lista de comprobación con otros probadores, el AP fomenta la consistencia y la colaboración y les ayuda a comprender mejor los elementos de prueba y las áreas críticas en las que concentrarse durante las pruebas.

3.4.3 Prueba multitudinaria

La prueba multitudinaria distribuye las pruebas entre un grupo de probadores internos o externos de diversa procedencia y ubicación. Puede ser una forma rentable de validar la usabilidad y abarca características de calidad tanto funcionales como no funcionales (Alyahya, 2020), (Leicht et al., 2017).

Algunas de las ventajas de la prueba multitudinaria son las siguientes:

- **Diversos entornos de prueba.**
  - Los probadores pueden estar en distintas ubicaciones geográficas y utilizar diversas configuraciones de entorno con una gran variedad de dispositivos, navegadores y condiciones de red.
- **Mayor flexibilidad.**
  - Fácilmente escalable para tratar muchas pruebas en poco tiempo.
- **Rentabilidad.**
  - La prueba multitudinaria suele ser menos costosa que mantener un equipo de prueba interno grande y diverso o complementarlo con servicios de prueba externos.

- **Retroalimentación rápida.**
  - Los probadores pueden proporcionar una retroalimentación rápida, lo que ayuda a identificar y solucionar los fallos de forma temprana.
- **Perspectiva del usuario real.**
  - Los probadores pueden ser usuarios reales de la aplicación y ofrecer una mejor perspectiva de su experiencia de usuario y usabilidad. Esto puede ser especialmente valioso en la prueba de aceptación de usuario.
- **Variabilidad.**
  - Como las pruebas son ejecutadas cada vez por una gran variedad de probadores, no son tan repetibles. Aunque esto podría ser una limitación, también da lugar a una cobertura más amplia, lo que aumenta las posibilidades de encontrar defectos.

Algunas de las limitaciones de la prueba multitudinaria son las siguientes:

- **Calidad poco fiable de la prueba.**
  - La calidad de la prueba puede variar significativamente en función de las habilidades de cada probador, aunque esto puede no ser relevante, por ejemplo, cuando el objetivo es la retroalimentación sobre la experiencia de usuario.
- **Retos de comunicación.**
  - La coordinación con muchos probadores de distintos lugares con diferentes husos horarios, diferencias culturales y barreras lingüísticas puede ser todo un reto.
- **Seguridad.**
  - Compartir software con probadores externos plantea riesgos de seguridad y confidencialidad de los datos. Unas medidas adecuadas pueden mitigar estos riesgos, permitiendo un uso responsable de la prueba multitudinaria sin revelar detalles sensibles ni facilitar el plagio.
- **Documentación y suministro de información.**
  - Asegurar una documentación detallada de las pruebas y gestionar un gran número de hallazgos, incluidos los duplicados y los falsos positivos, puede ser todo un reto cuando se trata de un grupo numeroso y diverso de probadores.

La prueba multitudinaria es un enfoque que no sustituye a los AP que aplican técnicas de prueba, pero que puede aumentar la cobertura de diversos entornos de prueba.

3.5 Aplicación de las técnicas de prueba más adecuadas

La prueba debe ser lo más eficaz y eficiente posible en un contexto determinado. Para ello, el AP apoya al jefe de prueba en la selección de la(s) técnica(s) de prueba más adecuada(s). Además, el AP puede utilizar la automatización para apoyar las actividades de prueba. Esto incluye la automatización del diseño de la prueba, descrita en esta sección, y el apoyo a la automatización de la ejecución de la prueba, descrita en la sección 1.3.6.

3.5.1 Selección de técnicas de prueba para mitigar riesgos de producto

La selección de la(s) técnica(s) de prueba más adecuada(s) es crucial para la efectividad y eficacia de la mitigación del riesgo de producto y en ella influyen una serie de factores, entre los que se incluyen los siguientes.

Los **objetivos de prueba** (véase ISTQB-CTFL, v4.0.1, sección 1.1.1) especifican qué aspectos del objeto de prueba deben evaluarse. Orientan la elección de las técnicas de prueba y dependen del tipo de sistema (por ejemplo, la prueba basada en el dominio para cálculos numéricos en ingeniería frente a la prueba de tabla de decisión en la gestión del riesgo crediticio).

Los **riesgos de producto** están asociados a defectos potenciales. Estos defectos pueden detectarse mejor utilizando técnicas de prueba concretas, ya que la mayoría de las técnicas de prueba se concentran en detectar tipos específicos de defectos (véanse las secciones 3.1, 3.2, 3.3 y 3.4 de este programa de estudio). Por ejemplo:

- Las técnicas de prueba basadas en datos pueden detectar defectos en el tratamiento de datos, la implementación de dominios, las interfaces de usuario, los cálculos y las combinaciones de parámetros.
- Las técnicas de prueba basadas en el comportamiento pueden detectar defectos en los requisitos de usuario, como características que faltan, defectos de comunicación y defectos de procesamiento.
- Las técnicas de prueba basadas en reglas pueden detectar defectos en la lógica y los flujos de control.

El análisis del riesgo también ayuda a determinar, por ejemplo, el enfoque de prueba adecuado:

- Criterios de salida basados en la cobertura. Cuanto mayor sea el nivel de riesgo, más rigurosa puede ser la cobertura requerida (por ejemplo, todas las combinaciones en la cobertura combinatoria en lugar de la cobertura por pares). Sin embargo, el AP debe tener siempre en cuenta el equilibrio entre la intensidad de la cobertura y el esfuerzo de prueba necesario.
- Las pruebas basadas en la experiencia pueden utilizarse cuando definir la cobertura es difícil, cuando el nivel de riesgo es bajo o cuando el calendario de proyecto es ajustado.

Base de prueba. Si la especificación del objeto de prueba se realiza mediante modelos, el AP puede utilizar técnicas de prueba basadas en esos modelos. Si es imposible o difícil obtener un oráculo de prueba a partir de la base de prueba, pueden aplicarse técnicas de prueba como la prueba metamórfica o técnicas de prueba basadas en la experiencia.

El **conocimiento de los tipos de defectos recurrentes** puede indicar la selección de las técnicas de prueba que se concentran en detectar esos defectos (por ejemplo, pruebas basadas en listas de comprobación). Si la expectativa es descubrir defectos similares a los de iteraciones o proyectos anteriores, puede ser razonable utilizar las mismas técnicas de prueba que han dado buenos resultados.

Conocimientos y experiencia del probador. Si el AP no está familiarizado con una técnica de prueba determinada, no es recomendable utilizarla en tareas de prueba críticas. El conocimiento del dominio también puede influir en la selección de las técnicas de prueba. Por ejemplo, un conocimiento escaso o nulo del dominio indica que técnicas de prueba como la prueba exploratoria pueden resultar ineficaces.

Ciclo de vida de desarrollo del software utilizado. Un modelo de desarrollo secuencial es propicio para emplear técnicas más formales. En cambio, un modelo de desarrollo iterativo puede ser más apropiado para adoptar técnicas de prueba ligeras (por ejemplo, técnicas de prueba basadas en la experiencia) o cuando el diseño de la prueba puede automatizarse.

Requisitos del cliente y contractuales. Los contratos pueden exigir explícitamente la realización de pruebas específicas (por ejemplo, en términos de determinados niveles de prueba o tipos de prueba), lo que influye en la selección de las técnicas de prueba (por ejemplo, los criterios de aceptación con un conjunto de escenarios proporcionados por el cliente sugieren el uso de una técnica de prueba basada en escenarios).

Requisitos normativos. Cuando un proyecto sigue un estándar, puede requerir el uso de técnicas de prueba específicas. Por ejemplo, la norma (ISO 26262, 2018) exige el uso de técnicas de prueba como la partición de equivalencias, el análisis de valores límite o la adivinación de errores, en función del ASIL («Automotive Safety Integrity Level») asignado a un objeto de prueba.

Las **restricciones del proyecto**, como el tiempo y el presupuesto, pueden afectar al uso de técnicas que consumen mucho tiempo o que requieren recursos de coste elevado.

Las técnicas de prueba suelen combinarse para aumentar la eficiencia y efectividad de la detección de defectos. Por ejemplo:

- El AVF puede utilizarse para las condiciones de guarda en las pruebas de transición de estado.
- La prueba de dominio puede utilizarse en la prueba basada en escenarios para determinar el valor de una condición a partir de una tabla de decisión o de una variable que se presente en un sistema sometido a prueba.
- Las pruebas basadas en escenarios pueden combinarse con la cobertura de decisión, una técnica de prueba de caja blanca de la que se habla en (ISTQB-TTA, v4.0), para cubrir rigurosamente las decisiones del proceso de negocio. Para ver un ejemplo, consulte la prueba de ciclo de proceso en (Koomen et al., 2006).
- Las pruebas basadas en escenarios pueden combinarse con la cobertura de trayectos circulares para abordar los riesgos específicos de las actividades cíclicas de los procesos de negocio.

3.5.2 Ventajas y riesgos de la automatización del diseño de pruebas

El AP puede utilizar herramientas para aplicar técnicas de prueba, especialmente técnicas de prueba de caja negra. Al automatizar el diseño de la prueba, el AP crea un modelo de prueba y genera automáticamente el producto de prueba a partir de ese modelo. Por ejemplo, el AP diseña un modelo de transición de estados y deja que una herramienta de prueba basada en modelos genere casos de prueba para la cobertura de ida y vuelta.

La automatización del diseño de la prueba suele mejorar la eficiencia y la efectividad de la prueba. Entre sus ventajas se incluyen:

- **Prevención de defectos.** El modelado para las pruebas es una forma eficaz de evaluar la calidad de la base de pruebas (véase el apartado 5.2.1).
- **Capacidad ampliada.** La automatización permite aplicar técnicas de prueba y criterios de cobertura más complejos, como las pruebas combinatorias, las pruebas aleatorias o la cobertura de conmutador de orden N, lo que reduce el riesgo de código no probado.
- **Comprensibilidad mejorada.** Los criterios de selección de pruebas especificados en una herramienta se refieren de forma más clara y visible a las condiciones de la prueba y justifican la cobertura generada de forma más comprensible.
- **Menos trabajo repetitivo.** El producto de prueba puede generarse a partir del modelo de prueba, lo que reduce el trabajo manual y repetitivo, como la especificación de las pruebas.

- **Menos esfuerzos de mantenimiento.** El modelo de prueba es la única fuente de verdad utilizada para obtener el producto de prueba. Como resultado, sólo se necesita mantener el modelo de prueba.
- **Menos productos de prueba defectuosos.** El trabajo manual es propenso a errores. El producto de prueba generado por las herramientas tiene una mayor calidad y consistencia.
- **Mayor colaboración en equipo.** Los implicados pueden revisar el modelo de pruebas para encontrar defectos o para comprender mejor las condiciones de prueba.
- **Trazabilidad mejorada.** Es más fácil vincular los elementos de un modelo de prueba a las condiciones de prueba que los propios casos de prueba. Si la herramienta lo permite, los casos de prueba generados heredarán esos vínculos, lo que mejorará la trazabilidad general en la prueba.
- **Varios formatos de salida.** Los productos de prueba pueden generarse en varios formatos de salida, según lo requieran otras herramientas y actividades posteriores.

El AP también debe tener en cuenta los riesgos de automatizar el diseño de las pruebas. Entre ellos se incluyen pasar por alto condiciones de prueba que no se muestran en el modelo, subestimar el esfuerzo de mantenimiento del modelo de prueba, que los implicados encuentren el modelo difícil de entender y los riesgos genéricos de la automatización de la prueba (véase ISTQB- CTFL, v4.0.1, Sección 6.2).

4 Prueba de características de calidad

Duración: 60 minutos

Palabras Clave[7]

En la siguiente tabla se presentan las palabras clave del capítulo. En este documento, se identifican dos tipos de palabras clave:

- **ISTQB**: identificarán palabras clave del proceso de prueba
- **ESPDOM**: identificarán palabras clave específicas de dominio

Tipo Palabra Clave	Español	Inglés
ISTQB	adaptabilidad	adaptability
ISTQB	compatibilidad	compatibility
ISTQB	flexibilidad	flexibility
ISTQB	pertinencia funcional	functional appropriateness
ISTQB	completitud funcional	functional completeness
ISTQB	corrección funcional	functional correctness
ISTQB	adecuación funcional	functional suitability
ISTQB	prueba funcional	functional testing
ISTQB	instalabilidad	installability
ISTQB	capacidad de interacción	interaction capability
ISTQB	interoperabilidad	interoperability
ISTQB	usabilidad	usability
ISTQB	experiencia de usuario	user experience

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

Objetivos de Aprendizaje para "Capítulo 4"

4.1 Prueba funcional

TA-4.1.1 (K2) Diferenciar entre prueba de corrección funcional, pertinencia funcional y completitud funcional.

4.2 Prueba de usabilidad

TA-4.2.1 (K2) Explicar cómo contribuye el analista de prueba a la prueba de usabilidad.

4.3 Prueba de flexibilidad

TA-4.3.1 (K2) Explicar cómo contribuye el analista de prueba a la prueba de adaptabilidad e instalabilidad.

4**.4 Prueba de compatibilidad**

TA-4.4.1 (K2) Explicar cómo contribuye el analista de prueba a la prueba de interoperabilidad.

Introducción a la prueba de características de calidad

Este programa de estudio utiliza el modelo de calidad del software proporcionado en la norma ISO 25010 (ISO/IEC 25010, 2023) como guía y analiza las características de calidad en las que se concentra un AP. Sin embargo, la terminología de usabilidad utilizada difiere de este estándar para ser consistente con el programa de estudio de Prueba de Usabilidad (ISTQB-UT, v1.0).

El apéndice F ofrece una visión general del modelo de calidad del estándar ISO 25010 actual, los cambios en comparación con la versión anterior y los programas de estudio ISTQB® relacionados que se concentran en probar características de calidad específicas.

4.1 Prueba funcional

La prueba funcional es una de las tareas centrales del AP. Mientras que (ISTQB-CTFL, v4.0.1) resume brevemente la prueba funcional como un tipo de prueba, este programa de estudio entra en más detalle, discutiendo las subcaracterísticas de la adecuación funcional.

4.1.1 Subcaracterísticas de la adecuación funcional

El modelo de calidad del producto de la norma ISO 25010 (ISO/IEC 25010, 2023) diferencia tres subcaracterísticas de la adecuación funcional. Aunque todas ellas pueden evaluarse mediante pruebas funcionales, no todas las actividades de pruebas funcionales las abordan igual de bien. El AP debe ser capaz de seleccionar los niveles de prueba y las técnicas de prueba adecuados para abordar los riesgos de producto relacionados con una subcaracterística específica de la adecuación funcional.

La **prueba de completitud funcional** debe abarcar todas las tareas especificadas del software y los objetivos de los usuarios previstos. La pregunta clave es si todo lo que se ha solicitado está implementado.

La completitud funcional debe abordarse lo antes posible mediante la revisión de la especificación de requisitos en los modelos de desarrollo secuencial y la discusión de las historias de usuario, incluidos los criterios de aceptación, durante la redacción colaborativa de historias de usuario en el desarrollo ágil de software. En la prueba de sistemas, la prueba de integración de sistemas y la prueba de aceptación, la completitud funcional puede comprobarse de forma dinámica.

Las técnicas de prueba basadas en el comportamiento, como la prueba basada en escenarios, son una buena opción, aunque también son adecuadas otras técnicas de prueba de caja negra. La trazabilidad entre la base de prueba, las condiciones de prueba y los casos de prueba es esencial a la hora de determinar el nivel de completitud funcional alcanzado.

La **prueba de corrección funcional** responde a la pregunta de si los resultados reales son correctos (por ejemplo, exactos, precisos y consistentes) para entradas válidas e inválidas. Es crucial encontrar un oráculo de prueba eficaz que proporcione los resultados esperados en detalle.

La corrección funcional puede probarse en cualquier nivel de prueba. En términos de desplazamiento a la izquierda, la mayor parte de la prueba de corrección funcional debería producirse en la prueba de componentes y en la prueba de integración de componentes. Incluso si el AP no es responsable de estos niveles de prueba, debería contribuir a ellos para alcanzar mejor los objetivos de prueba.

Todas las técnicas de prueba de caja negra, las técnicas de prueba basadas en la experiencia y la prueba basada en la colaboración son adecuadas.

La prueba de pertinencia funcional verifica que las funciones facilitan la realización de las tareas y los objetivos especificados. Se concentra en si todo lo implementado satisface las necesidades de los usuarios.

La prueba de pertinencia funcional puede incluir revisiones del diseño de la interfaz de usuario, especialmente en el caso de las aplicaciones interactivas. La prueba dinámica comienza con la prueba de sistema y la prueba de aceptación en los modelos de desarrollo secuencial, así como con sesiones de demostración en el desarrollo ágil de software.

La prueba exploratoria y la prueba basada en la colaboración son las más apropiadas. Además, las técnicas de prueba de caja negra basadas en el comportamiento también son adecuadas.

4.2 Prueba de usabilidad

La usabilidad se refiere a un concepto amplio de características de calidad relacionadas con el usuario, que abarca la capacidad de interacción del modelo de calidad del producto (ISO/IEC 25010, 2023) y la utilidad del modelo de calidad de uso (ISO/IEC 25019, 2023). Puede encontrar más información sobre la prueba de usabilidad en (ISTQB-UT, v1.0), (ISO 9241-210, 2019) y (UXQB-FL, v4.01).

4.2.1 Contribución del analista de prueba a la prueba de usabilidad

En general, la prueba de usabilidad se concentra en la evaluación de los siguientes aspectos:

- capacidad de interacción - permitir a los usuarios completar tareas en contextos de uso específicos (ISO/IEC 25010, 2023) de forma eficaz, eficiente y satisfactoria.
- experiencia de usuario - abordar las percepciones de los usuarios antes, durante y después de interactuar con el objeto de prueba.
- accesibilidad - asegurar que los usuarios con discapacidades, diversos orígenes culturales o barreras lingüísticas puedan utilizar el sistema de forma eficaz y eficiente.

El AP puede colaborar en la prueba de usabilidad desde una fase temprana utilizando sus conocimientos sobre los grupos de usuarios objetivo, sus objetivos, su contexto de uso, las posibles dificultades para utilizar el sistema o la experiencia negativa de los usuarios.

El AP puede contribuir a las principales técnicas de evaluación de usabilidad de la siguiente manera:

- Las **revisiones de usabilidad** son realizadas por expertos en usabilidad para identificar posibles problemas de usabilidad y desviaciones de los criterios establecidos. Las revisiones de usabilidad pueden variar desde revisiones informales hasta inspecciones. El AP puede adaptar los criterios de revisión a las necesidades específicas de los grupos de usuarios, a los objetivos y prioridades particulares del negocio y al contexto de uso (por ejemplo, confeccionando una lista de comprobación de usabilidad genérica).
- Las **sesiones de prueba de usabilidad** implican a futuros usuarios o a sus representantes intentando resolver tareas predefinidas para evaluar si las tareas pueden completarse de forma eficaz, eficiente y satisfactoria. El AP puede contribuir a diseñar escenarios para las sesiones de prueba de usabilidad en función de personas, grupos de usuarios o perfiles operativos.
- Los **cuestionarios o encuestas a los usuarios** que incluyen valoraciones y retroalimentaciones miden la satisfacción de los usuarios. Ejemplos de cuestionarios para usuarios son el IMUS (Inventario de Medición de Usabilidad Software IMUS, 1991) y el IAMSW (Inventario de Análisis y Medición de Sitios Web IAMSW, 1999). El AP puede ayudar a diseñar un cuestionario y evaluar las respuestas para abordar los objetivos específicos de los usuarios a los que va dirigido dentro de su contexto de uso.

En la prueba de accesibilidad, un objetivo de prueba común es verificar la conformidad con los estándares. El estándar internacional Directrices de Accesibilidad al Contenido Web (WCAG, 2023) define tres niveles

de conformidad (A, AA y AAA) que representan grados crecientes de accesibilidad al contenido Web. Los estándares nacionales incluyen la Ley de Igualdad del Reino Unido (Gobierno del Reino Unido, 2010) y la Ley de Discapacidades de los Estadounidenses (Departamento de Justicia de EE.UU., 2010). El AP puede identificar el nivel de conformidad requerido y las necesidades específicas del grupo destinatario analizando el contexto de uso.

4.3 Prueba de flexibilidad

La prueba de flexibilidad (también conocida como prueba de portabilidad) verifica que el objeto de prueba puede adaptarse a los cambios en sus contextos de uso o en el entorno de sistemas. El modelo de calidad del producto ISO 25010 (ISO/IEC 25010, 2023) distingue entre las siguientes subcaracterísticas de flexibilidad:

- adaptabilidad.
- escalabilidad.
- instalabilidad.
- capacidad de ser reemplazado.

La flexibilidad tiene aspectos tanto técnicos como no técnicos. Esta sección se concentra en cómo el AP puede contribuir a las pruebas de adaptabilidad e instalabilidad. La escalabilidad y la capacidad de ser reemplazado están relacionadas con aspectos técnicos y se tratan en (ISTQB-PT, v1.0) y (ISTQB-TTA, v4.0), respectivamente.

4.3.1 Contribución del analista de prueba a la prueba de adaptabilidad y a la prueba de instalabilidad

La prueba de adaptabilidad verifica que el objeto de prueba puede adaptarse o transferirse al hardware, software u otros entornos operativos o de uso previstos.

El AP apoya la prueba de adaptabilidad identificando los entornos de destino previstos (por ejemplo, las versiones de los sistemas operativos móviles compatibles y las versiones de los navegadores que pueden utilizarse) y diseñando pruebas que cubran combinaciones de estos entornos. Dado que esto requiere datos de prueba que representen varias configuraciones de parámetros del entorno, a menudo se aplican técnicas de prueba como la prueba combinatoria (véase el apartado 3.1.2). Otro ejemplo es la integración de varios componentes de la plataforma en los proyectos de los clientes para asegurar la compatibilidad en todos los entornos de destino. En función del riesgo de producto, el AP diseña y ejecuta pruebas de humo o un juego de pruebas más completo para verificar que el objeto de prueba se adapta correctamente al entorno de destino.

Siguiendo las buenas prácticas de la prueba de adaptabilidad (por ejemplo, definiendo de forma temprana los entornos de destino, utilizando la prueba combinatoria, la prueba de humo en nuevos entornos y la monitorización de defectos específicos del entorno), el AP puede descubrir defectos que podrían limitar la vida útil y la usabilidad del software (por ejemplo, la facilidad de uso en diferentes tamaños de pantalla). El trabajo del AP en la prueba de adaptabilidad también debe estar respaldado por la prueba automatizada multiplataforma implementada por ingenieros de automatización de la prueba. Una prueba de adaptabilidad rigurosa puede detectar a tiempo dificultades específicas del entorno, lo que ayuda a evitar actualizaciones o rediseños importantes y frecuentes tras el despliegue y a reducir los costes de mantenimiento.

La prueba de instalabilidad verifica que el objeto de prueba puede instalarse, desinstalarse, actualizarse y reconfigurarse correctamente en los entornos especificados. Las pruebas de instalabilidad van más allá de la mera comprobación de si el procedimiento de instalación se ejecuta hasta su compleción.

Los objetivos típicos de prueba de instalabilidad en los que se concentra el AP son:

- Verificar que los procedimientos de instalación se ejecutan correctamente bajo diversas configuraciones de parámetros del entorno. Esto hace que resulten útiles técnicas de prueba como la prueba combinatoria, similar a la prueba de adaptabilidad.
- Diseñar y ejecutar pruebas para determinar si el objeto de prueba funciona correctamente tras su instalación o actualización.
- Comprobar la facilidad con la que los usuarios instalan, desinstalan o actualizan el software. Esto incluye la revisión de la documentación de instalación.
- Probar el comportamiento relacionado con los permisos, especialmente en el caso de las aplicaciones móviles (véase ISTQB-MAT, v1.0).

4.4 Prueba de compatibilidad

La prueba de compatibilidad verifica si un objeto de prueba es compatible con otros componentes o sistemas cuando se utiliza. El modelo de calidad de producto ISO 25010 (ISO/IEC 25010, 2023) diferencia entre dos subcaracterísticas de compatibilidad: interoperabilidad y coexistencia. Por lo tanto, las pruebas de compatibilidad pueden subdividirse en los siguientes tipos de pruebas:

La prueba de interoperabilidad, que verifica la compatibilidad con los componentes o sistemas con los que está previsto que interactúe el objeto de prueba. Estas pruebas suelen ser pruebas funcionales de caja negra, por lo que el AP suele encargarse de ellas.

La prueba de coexistencia, que verifica que el objeto de prueba puede compartir su entorno de destino con otros componentes o sistemas sin interferencias. Este tipo de prueba técnica se trata en el programa de estudio del Analista de Pruebas Técnicas (ISTQB-TTA, v4.0).

4.4.1 Contribución del analista de pruebas a la prueba de interoperabilidad

El objetivo de las pruebas de interoperabilidad es comprobar que dos o más componentes o sistemas pueden intercambiar información y utilizar mutuamente la información que se ha intercambiado. Cuando el intercambio de información implica una transformación de datos, la prueba de interoperabilidad debe incluir la verificación de dicha transformación.

La prueba de interoperabilidad es especialmente importante cuando varios sistemas necesitan colaborar, compartir datos o realizar tareas juntos. Esto ocurre especialmente en las arquitecturas de software modernas, como las soluciones en la nube, los servicios web, los microservicios, la contenerización e Internet de las cosas.

La interoperabilidad suele darse en varios niveles arquitectónicos. El AP debe comprender las posibles interacciones para definir las condiciones de prueba adecuadas para cubrirlas. Puede que no todas las interacciones estén documentadas. El AP puede recuperar indirectamente información sobre las interacciones de la documentación de arquitectura y diseño. Por lo tanto, es crucial comprender esta documentación para asegurar que se probarán todos los aspectos importantes de las interacciones.

La prueba de interoperabilidad puede detectar defectos en:

- las transformaciones de datos para el intercambio de datos.
- la interpretación o el uso de los datos intercambiados.

- flujos y protocolos de comunicación.
- conformidad con los estándares.
- la funcionalidad de extremo a extremo.
- documentación de diseño.

La prueba de interoperabilidad suele aplicarse durante la prueba de integración. Las técnicas de prueba de caja negra, como las técnicas de prueba basadas en datos que se concentran en los datos intercambiados, las técnicas de prueba basadas en el comportamiento para interpretar o utilizar los datos, y las técnicas de prueba de funcionalidad de extremo a extremo o basadas en reglas para los datos para la transformación de datos, son muy adecuadas para la prueba de interoperabilidad. Las técnicas de prueba basadas en la experiencia pueden ser un complemento útil de las pruebas de caja negra. Los casos de prueba de las pruebas funcionales o de adaptabilidad pueden reutilizarse para la prueba de interoperabilidad.

Ejemplos de estándares generales en interoperabilidad son (ISO 15745, 2003) e (ISO 16100, 2009). El ETSI (ETSI EG 202 237 v1.2.1, 2010) ofrece un ejemplo de metodología concreta de pruebas de interoperabilidad en el dominio de las telecomunicaciones.

5 Prevención de defectos de software

Duración: 225 minutos

Palabras Clave[8]

En la siguiente tabla se presentan las palabras clave del capítulo. En este documento, se identifican dos tipos de palabras clave:

- **ISTQB**: identificarán palabras clave del proceso de prueba
- **ESPDOM**: identificarán palabras clave específicas de dominio

Tipo Palabra Clave	Español	Inglés
ISTQB	revisión ad hoc	ad hoc reviewing
ISTQB	revisión basada en lista de comprobación	checklist-based reviewing
ISTQB	prevención de defectos	defect prevention
ISTQB	prueba basada en modelos	model-based testing
ISTQB	lectura basada en perspectiva	perspective-based reading
ISTQB	técnica de revisión	review technique
ISTQB	revisión basada en roles	role-based reviewing
ISTQB	análisis de la causa raíz	root cause analysis
ISTQB	revisión basada en escenarios	scenario-based reviewing
ISTQB	resultado de prueba	test result

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

Objetivos de Aprendizaje para "Capítulo 5"

5.1 Prácticas de prevención de defectos

TA-5.1.1 **(K2)** Explicar cómo el analista de prueba puede contribuir a la prevención de defectos.

5.2 Apoyo a la contención de fase

TA-5.2.1 **(K3)** Utilizar un modelo del objeto de prueba para detectar defectos en una especificación.

TA-5.2.2 **(K3)** Aplicar una técnica de revisión a una base de prueba para encontrar defectos.

5.2 Mitigar la recurrencia de defectos

TA-5.3.1 **(K4)** Analizar los resultados de pruebas para identificar posibles mejoras en la detección de defectos.

TA-5.3.2 **(K2)** Explicar cómo la clasificación de defectos apoya el análisis de la causa raíz.

Introducción a la prevención de defectos de software

El objetivo de la prevención de defectos es la implementación de acciones que reduzcan la probabilidad de (re)aparición de defectos en los productos de trabajo y mitiguen la propagación de los defectos a las fases posteriores del ciclo de vida de desarrollo de software (CVDS). Estos esfuerzos dan como resultado una serie de beneficios importantes, como la reducción de costes y mano de obra, el aumento de la productividad y la mejora de la calidad del producto. La prevención de defectos es responsabilidad de todo el equipo. El AP puede utilizar sus conocimientos y experiencia específicos para contribuir a ello.

Las prácticas de prevención de defectos incluyen:

- evitar la introducción de defectos, que forma parte de las actividades de aseguramiento de la calidad.
- evitar que los defectos escapen a fases posteriores del ciclo de vida de desarrollo de software (CVDS) (véase la sección 5.2).
- evitar que los defectos se repitan (véase la Sección 5.3).

5.1 Prácticas de prevención de defectos

5.1.1 Contribución del analista de prueba a la prevención de defectos

El AP puede contribuir a la prevención de defectos de varias maneras, utilizando sus conocimientos del dominio, su experiencia en materia de prueba y sus habilidades analíticas. Algunos ejemplos son:

- Participación en el análisis del riesgo - asegurando que los riesgos identificados se mitigan adecuadamente (por ejemplo, seleccionando las técnicas de prueba más adecuadas).
- Revisión de requisitos, modelos y especificaciones - permitiendo la detección temprana de defectos en la base de prueba evitando que se escapen y lleguen al código, lo que disminuye significativamente el coste de solucionarlos.
- Participación en retrospectivas - permitiendo la identificación de posibles mejoras en el análisis de prueba, el diseño de prueba, la implementación de prueba y la ejecución de prueba (por ejemplo, utilizando técnicas de prueba más eficaces, dirigiendo la prueba a áreas específicas de riesgo o mejorando los datos de prueba y los entornos de prueba para reducir tanto los resultados falsos positivos como los falsos negativos) para evitar mejor que se escapen los defectos.
- Recopilación y evaluación de datos sobre defectos - apoyando el análisis de la causa raíz y la mejora del proceso mediante la recopilación de datos detallados sobre los defectos para posibilitar su clasificación y análisis estadístico y facilitar así el análisis de la causa raíz.
- Participación en el análisis de la causa raíz - evitar que los defectos vuelvan a producirse proponiendo acciones correctivas para abordar las causas raíz identificadas.

Además de participar en la prevención de defectos, el AP evalúa (normalmente en colaboración con el jefe de la prueba) si las medidas propuestas han dado el resultado deseado. Algunos ejemplos de métricas que ayudan a evaluar la efectividad de estas medidas son:

- **Eficiencia de eliminación de defectos (EED):** mide la proporción de defectos eliminados antes de la entrega y el número total de defectos. El denominador (desconocido) puede estimarse o sustituirse por el número total de defectos encontrados hasta un momento determinado (por ejemplo, hasta 6 meses después de la entrega). Una EED alto indica que se escapan menos defectos a producción, lo que podría implicar mejores prácticas de prevención de defectos. Sin embargo, la EED no distingue entre defectos detectados mediante prevención y detección.

- **Efectividad de la contención de fase (ECF):** mide el número de defectos introducidos y eliminados en la misma fase en relación con el número total de defectos introducidos en esa fase. Una ECF elevada indica que se escapan menos defectos a fases posteriores.
- **Coste de la calidad** - ilustra la relación entre los costes de prevención de defectos, de detección de defectos y de eliminación (véase ISTQB-TM, v3.0, Sección 3.2.1).

5.2 Apoyo a la contención de fase

El objetivo de la contención de fase es detectar y eliminar defectos en la misma fase del ciclo de vida de desarrollo de software (CVDS) en la que se introdujeron. En el Desarrollo Ágil de Software, el enfoque de desplazamiento a la izquierda puede utilizarse de forma similar.

Esta política reduce el coste de la calidad. Dado que la base de prueba es una entrada importante para el análisis de prueba y el diseño de prueba, el AP puede contribuir mejor a la contención de fase evaluando la calidad de la base de prueba.

Prestar atención a la calidad de la base de prueba de forma temprana minimizará el esfuerzo posterior y evitará la propagación de defectos a fases posteriores. Este programa de estudio analiza dos opciones habituales para que el AP encuentre defectos en la base de prueba: el modelado a efectos de prueba y la revisión de la base de prueba mediante diversas técnicas de revisión (ISO/IEC 20246, 2017).

5.2.1 Uso de modelos para detectar defectos

El modelado proporciona abstracciones de un sistema a un cierto nivel de precisión y detalle. Ayuda a los implicados a comprender mejor el sistema que se está desarrollando. Como se explica a continuación, el modelado puede apoyar la contención de fase al menos de tres maneras:

- detectando defectos en las especificaciones.
- detectando defectos en los modelos.
- detectando defectos en objetos de prueba mediante pruebas basadas en modelos.

Detección de defectos en las especificaciones. Las especificaciones suelen proporcionarse en forma de texto escrito de manera informal. El AP puede representar formalmente estas especificaciones mediante modelos. Al crear un modelo, las condiciones de prueba (por ejemplo, los requisitos) se asignan a los elementos del modelo y se vinculan a ellos para su trazabilidad. La formalización y la visualización de las condiciones de prueba en un modelo revelan eficazmente defectos como el carácter incompleto, las incoherencias o las ambigüedades. El punto fuerte de la modelización es que la especificación se transforma mientras que las revisiones la comprueban en su forma original. Como resultado, el AP también contribuye a encontrar soluciones adecuadas para los defectos detectados.

Los modelos basados en datos incluyen modelos de dominio y modelos de prueba combinatoria. Permiten al AP detectar defectos de dominio como particiones solapadas, vacíos en la cobertura del dominio, particiones vacías o combinaciones de parámetros inexistentes o incorrectas.

Los modelos basados en el comportamiento incluyen las matrices CLAB, los modelos basados en estados o los modelos de escenarios. Permiten al AP detectar ciclos de vida de entidades incompletos o incoherentes, transiciones de estado omitidas o defectuosas, bloqueos mutuos, bucles sin fin, comportamientos del sistema ambiguos o incoherentes o falta de tratamiento de excepciones.

Los modelos basados en reglas, como las tablas de decisión o las relaciones metamórficas, permiten al AP detectar defectos en las reglas de negocio, como omisiones, incoherencias, ambigüedades, redundancias, escenarios propensos a errores o lógica de negocio compleja.

El modelado también puede detectar defectos como incoherencias en la nomenclatura, los valores de los datos (por ejemplo, las fronteras) y las entradas o salidas. También puede identificar información faltante, incompleta, ambigua o innecesaria.

Detección de defectos en los modelos. Si la base de prueba contiene modelos, el AP puede analizarlos para detectar defectos. Este programa de estudio analiza la detección de defectos en tres modelos típicos utilizados en las pruebas de software: diagramas de transición de estados (véase el apartado 3.2.2), diagramas de actividades (véase el apartado 3.2.3) y tablas de decisión (véase el apartado 3.3.1). Entre los ejemplos de defectos en los modelos mencionados se incluyen:

- diagramas de transición de estados - estados ausentes/incorrectos, transiciones inadecuadas, condiciones o acciones de guarda incorrectas, estados redundantes o inalcanzables y comportamiento no determinista.
- diagramas de actividad - acciones que faltan, inalcanzables o sin salida, orden incorrecto de las acciones, condiciones de guarda incorrectas, no excluyentes o incompletas en los nodos de decisión, puntos de sincronización que faltan o flujos paralelos mal sincronizados que pueden dar lugar a un comportamiento no intencionado.
- tablas de decisión: reglas solapadas, incoherentes o inviables, incompletitud (por ejemplo, combinaciones de condiciones que faltan o acciones que faltan para una determinada combinación de condiciones).

Todos los modelos pueden contener también defectos como errores de sintaxis, erratas, duplicados y una nomenclatura incoherente de los elementos del modelo.

Detección de defectos mediante pruebas basadas en modelos (PBM). En este enfoque de prueba, una herramienta de PBM diseña y genera casos de prueba y otros productos de prueba basados en un modelo PBM y en criterios de selección de pruebas definidos por el AP. PBM es eficaz para encontrar anomalías en la especificación porque permite una cobertura exhaustiva y una exploración sistemática del comportamiento previsto del objeto de prueba según el modelo PBM. Los Modelos PBM, especialmente los gráficos, fomentan la comunicación con los implicados, creando una percepción y comprensión comunes de la base de prueba. Encontrará más detalles sobre PBM en el programa de estudio del probador basado en modelos (ISTQB-PBM, v1.1).

5.2.2 Aplicación de técnicas de revisión

Una base de pruebas bien definida es la piedra angular para asegurar la calidad de los productos del trabajo y el éxito de los proyectos. La revisión de la base de prueba ayuda a identificar y abordar los defectos de forma temprana, evitando que escapen a las fases posteriores.

El AP puede emplear diversas técnicas de revisión durante la revisión individual para identificar defectos en la base de prueba. La selección de la técnica de revisión más adecuada puede aumentar la eficiencia y la efectividad de la revisión. Esta selección debe tener en cuenta factores como las metas de la revisión, los objetivos del proyecto, los recursos disponibles, el tipo de base de prueba, los riesgos asociados, el dominio del negocio y la cultura de la empresa.

A continuación, este programa de estudio analiza cinco técnicas de revisión utilizadas habitualmente por el AP.

- **Revisión ad hoc**
  - La **revisión ad hoc** la llevan a cabo los revisores de manera informal, sin un proceso estructurado. Los revisores reciben poca o ninguna orientación sobre cómo debe realizarse la tarea. La revisión ad hoc necesita poca preparación y depende en gran medida de las habilidades de los revisores. Durante la revisión, los revisores leen la base

de prueba y documentan las anomalías a medida que las encuentran. Esta técnica, si no se gestiona, puede dar lugar a un gran volumen de informes duplicados de anomalías procedentes de múltiples revisores.

- **Revisión basada en listas de comprobación**
  - La **revisión basada en listas de comprobación** consiste en evaluar la base de prueba con respecto a una lista de comprobación predefinida. Las listas de comprobación recuerdan a los revisores que deben comprobar puntos específicos y pueden despersonalizar la revisión. Las listas de comprobación pueden ser genéricas o específicas de las características de calidad, los objetivos de prueba o el tipo de base de prueba. El AP adapta la lista de comprobación al tipo de base de prueba, al nivel de riesgo o a la condición de prueba. Esto asegura que la revisión se concentrará en los aspectos más relevantes de la base de prueba. Las listas de comprobación deben actualizarse periódicamente con los defectos pasados por alto. Mantener las listas de comprobación actualizadas evita pasar por alto anomalías recién identificadas. Las listas de comprobación no son exhaustivas.

    Por lo tanto, el AP no se limita a comprobar los elementos de la lista. Esto maximiza la detección de defectos y permite al AP capturar anomalías que las listas de comprobación pueden no cubrir explícitamente.
- **Revisión basada en escenarios**
  - La **revisión basada en escenarios** consiste en simular un proceso o una actividad para identificar anomalías y perfeccionar la base de la prueba. Esta técnica de revisión es más eficaz cuando la base de prueba tiene un formato basado en escenarios, como un diagrama de casos de uso o de actividades. En estos casos, los revisores pueden realizar «simulacros» basados en el uso previsto del producto de trabajo. Los escenarios proporcionan directrices valiosas, pero los revisores no están limitados a los escenarios documentados y también deben pensar más allá de los escenarios para identificar más anomalías.
- **Revisión basada en roles**
  - La **revisión basada en roles implica** la asignación de roles o responsabilidades específicas a los revisores. Los roles típicos se basan en tipos específicos de usuario final (por ejemplo, usuario experimentado, inexperto, senior o infantil) o roles específicos en la organización (por ejemplo, administrador o usuario habitual). Cada rol puede describirse mediante un personaje (es decir, el personaje concreto pero ficticio diseñado para representar las características, necesidades, objetivos y preferencias de un grupo concreto de usuarios). Al distribuir las responsabilidades entre los revisores en función de sus roles, las revisiones basadas en roles permiten a los individuos concentrarse en aspectos específicos de la base de prueba, asegurando una cobertura en profundidad y, al mismo tiempo, evitando la duplicación de anomalías.
- **Lectura basada en perspectiva**
  - La **lectura basada en perspectiva** implica revisar la base de la prueba desde varias perspectivas o puntos de vista (por ejemplo, diseñador, probador, comercial, administrador y usuario final). Esto conduce a una revisión individual más profunda con menos duplicación de anomalías entre los revisores. Además, la lectura basada en perspectiva requiere que los revisores intenten utilizar la base de prueba objeto de revisión para generar el producto de trabajo que obtendrían de ella. Por ejemplo, un probador intentaría

generar borradores de pruebas de aceptación basados en especificaciones de requisitos para ver si se incluye toda la información necesaria.

5.3 Mitigación de la recurrencia de defectos

Un AP puede ayudar proactivamente a minimizar la recurrencia de defectos en el software. Este programa de estudio trata dos enfoques relacionados con la mitigación de la recurrencia de defectos: el análisis de resultados de prueba para mejorar el análisis de la prueba y el diseño de prueba y el apoyo al análisis de la causa raíz con la clasificación de defectos.

5.3.1 Análisis de los resultados de las pruebas para mejorar la detección de defectos

Los resultados de las pruebas permiten al AP identificar fallos, pero también le proporcionan retroalimentación para ayudar a mejorar la efectividad de la detección de defectos. A continuación, se describen algunas técnicas de uso común para analizar los resultados de las pruebas.

- **Análisis de agrupamiento de defectos previstos frente a los reales.**
  - Unos pocos componentes suelen contener la mayoría de los defectos (véase ISTQB-CTFL, v4.0.1, sección 1.3). Tras la prueba, el AP puede predecir las zonas propensas a los defectos y comparar los agrupamientos de defectos predichos frente a los reales. En caso de discrepancias, pueden aplicarse pruebas más rigurosas a las zonas en las que se hayan encontrado más defectos de los previstos. A la hora de determinar los agrupamientos, los criterios medibles deben asegurar la claridad y la consistencia (por ejemplo, la densidad de defectos y la severidad de los defectos). Entre estos criterios, la severidad de los defectos debe desempeñar un papel importante. Los pequeños agrupamientos de defectos críticos o importantes suelen ser más importantes (es decir, necesitan pruebas más rigurosas) que los grandes agrupamientos de defectos menores o cosméticos.
- **Análisis del porcentaje de detección de defectos (PDD).**
  - El PDD es una de las medidas de efectividad de la prueba más importantes para un nivel de prueba. Al calcular el PDD, el número de defectos fugados debe limitarse a los que el nivel de prueba considerado podría haber detectado. Para asegurar la consistencia, deben establecerse fronteras claras para el recuento de defectos, como cronologías (por ejemplo, defectos encontrados en un plazo definido tras la entrega) y criterios de exclusión (por ejemplo, defectos en componentes de terceros o en entornos específicos de clientes). Un PDD bajo para un nivel de prueba determinado indica un alto porcentaje de defectos fugados, lo que significa que la detección de defectos es ineficaz. En tal caso, el AP debe analizar las razones y proponer medidas para mejorarlo, haciendo que el nivel de prueba sea más concentrado y riguroso. Lo mejor es dividir el PDD por niveles de severidad, ya que la prioridad de reducir los defectos fugados suele depender de su severidad.
- **Análisis de cobertura estructural**
  - El análisis de cobertura estructural evalúa hasta qué punto las pruebas han practicado áreas específicas del objeto de prueba. Identificar las áreas de baja cobertura permite al AP dirigir los esfuerzos de prueba a esas áreas. Aumentar su cobertura ayuda a descubrir nuevos defectos fugados anteriormente. La cobertura estructural, como la cobertura de sentencias, la cobertura de ramas o la cobertura de neuronas, suele medirse con

herramientas de prueba. A la hora de seleccionar las áreas adicionales que deben cubrirse, deben tenerse en cuenta sus niveles de riesgo.

- **Análisis de divergencia de la prueba evalúa**
  - El análisis de divergencia de la prueba evalúa hasta qué punto las pruebas han practicado los cambios de código recientes. Esto permite al AP concentrar el esfuerzo de prueba adicional en las áreas especialmente propensas a errores (es decir, los nuevos cambios que no se han probado en absoluto) en lugar de dirigirse a todas las áreas que tienen una baja cobertura (por ejemplo, el código que no ha cambiado en mucho tiempo y que se ha probado en entregas anteriores).
- **Análisis del patrón de llegada de defectos.**
  - El número o la densidad de defectos encontrados en fases sucesivas de un proyecto (por ejemplo, iteraciones) pueden compararse con patrones que describen la distribución teórica de estos valores a lo largo del tiempo. Un ejemplo clásico de patrón de llegada de defectos es el modelo de Rayleigh (Elsayed, 2021). Tiene un único pico y está sesgado hacia la derecha, lo que muestra que el número esperado de defectos encontrados aumenta primero en el tiempo y, tras alcanzar su valor máximo, desciende lentamente hacia cero. Basándose en el análisis de dicho patrón, es posible inferir la fuerza de los casos de prueba existentes y el potencial para su mejora. Por ejemplo, supongamos que la detección de defectos se mantiene en un nivel bajo y constante cuando el patrón sugiere que debería aumentar. En ese caso, puede significar que las pruebas existentes son demasiado débiles e incapaces de detectar defectos adicionales.

Los métodos analíticos descritos anteriormente utilizan varias métricas relacionadas con los defectos, como el número de defectos o DDP. Estas métricas se calculan a partir de los resultados de las pruebas (por ejemplo, el número de pruebas pasadas/fallidas), los informes de defectos y las métricas estructurales (por ejemplo, la cobertura de código o la densidad de defectos). Sin embargo, es preciso tener en cuenta que estas métricas no siempre son tan fáciles de leer a partir de los resultados de prueba como puede parecer. Por ejemplo, el número de pruebas fallidas no es necesariamente el mismo que el número de defectos detectados por estas pruebas. Para calcular el número real de defectos detectados, hay que analizar cuidadosamente los resultados del proceso de depuración porque la relación entre los resultados de las pruebas y los defectos puede ser de muchos a muchos. Varias pruebas pueden detectar el mismo defecto o una sola prueba puede detectar varios defectos. Además, la severidad de los defectos puede diferir de la criticidad de las pruebas, ya que un caso de prueba crítico puede fallar debido a un defecto cosmético.

5.3.2 Apoyo al análisis de la causa raíz con la clasificación de defectos

El análisis de la causa raíz (ACR) es una técnica para identificar y abordar las causas subyacentes o fundamentales de un defecto en lugar de sólo sus síntomas. El ACR respalda un enfoque estructurado de la mejora de la calidad y su objetivo principal es evitar que se repitan los defectos. El ACR utiliza diversas técnicas para identificar las causas raíz de los defectos y los fallos (por ejemplo, taxonomías de defectos, la técnica de los cinco porqués, diagramas causa-efecto y análisis de Pareto).

El ACR clásico implica que los expertos en la materia estudien un defecto con bastante detalle después de resolverlo. Sin embargo, suele haber muchos defectos que necesitan ser analizados. Por lo tanto, sería muy ineficaz y llevaría mucho tiempo planificar acciones preventivas para cada defecto. Una forma de abordar este problema es clasificar los defectos y, a continuación, realizar el ACR para los tipos de defecto que se produzcan.

La clasificación de defectos se basa en el reconocimiento de que los defectos individuales capturan una gran cantidad de información sobre el proceso de desarrollo y el sistema sujeto a prueba. La clasificación

de defectos permite al AP extraer del defecto información sobre diversos aspectos del proceso de desarrollo y convertirla en una medida del proceso. Esto, a su vez, permite conocer los tipos de errores cometidos durante el desarrollo, lo que resulta útil para la mejora del proceso. La clasificación de defectos salva la divergencia entre las estadísticas cuantitativas de defectos y el ACR cualitativo. Para apoyar eficazmente el ACR, los defectos deben clasificarse uniformemente a lo largo de todo el ciclo de vida de desarrollo de software (CVDS), desde las pruebas tempranas hasta la producción.

El AP debe apoyar a su organización en la estandarización de la clasificación de defectos del software. Esto mejorará la comunicación y el intercambio de información sobre defectos entre desarrolladores y organizaciones, facilitando el ACR.

Algunos ejemplos de métodos de clasificación de defectos son:

- la clasificación ortogonal de defectos (COD) (Chillarege, 1992), que clasifica cada defecto en atributos ortogonales (es decir, mutuamente excluyentes), recogidos tanto cuando se informa de un defecto como cuando se soluciona
- IEEE 1044 (IEEE 1044, 2009), una clasificación estándar para anomalías de software que proporciona un conjunto básico de atributos para la clasificación de fallos y defectos
- clasificación basada en la severidad, que clasifica los defectos en función de su severidad (por ejemplo, crítico, mayor, menor, trivial)
- modelos de taxonomía de defectos, como (Beizer, 1990) o (Catolino et al., 2019)

Los defectos también pueden asignarse a atributos de calidad utilizando modelos de calidad del software, como (ISO/IEC 25010, 2023) o el modelo FURPS (Grady et al., 1987).

Se puede encontrar más información sobre RCA en (ISTQB-ITP, v1.0).

6 Referencias

6.1 Estándares

- ETSI EG 202 237 v1.2.1: Internet Protocol Testing (IPT), 2010. Standard. European Telecommunications Standards Institute.
- IEEE 1044: Standard Classification for Software Anomalies, 2009. Standard. Institute of Electrical and Electronics Engineers.
- ISO 15745: Industrial automation systems and integration – Open systems application integration framework, 2003. Standard. International Organization for Standardization.
- ISO 16100: Industrial automation systems and integration – Manufacturing software capability profiling for interoperability, 2009. Standard. International Organization for Standardization.
- ISO 26262: Road vehicles – functional safety – Part 6: Product development at the software level, 2018. Standard. International Organization for Standardization.
- ISO 9241-210: Ergonomics of human-system interaction, part 210: Human-centred design for interactive systems, 2019. Standard. International Organization for Standardization.
- ISO/IEC 20246: Software and systems engineering – Work product reviews, 2017. Standard. International Organization for Standardization.
- ISO/IEC 25010: Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality model, 2023. Standard. International Organization for Standardization.
- ISO/IEC 25019: Systems and software Quality Requirements and Evaluation (SQuaRE) – Quality-in-use model, 2023. Standard. International Organization for Standardization.
- ISO/IEC/IEEE 29119-3: Software testing, Part 3: Test documentation, 2021. Standard. International Organization for Standardization.
- ISO/IEC/IEEE 29119-4: Software testing, Part 4: Test techniques, 2021. Standard. International Organization for Standardization.
- ISO/IEC/IEEE 29119-5: Software testing, Part 5: Keyword-Driven Testing, 2016. Standard. International Organization for Standardization.
- OMG® BPMN: Business Process Model and Notation, version 2.0, 2013. Standard. Object Management Group, OMG®.
- OMG® DMN: Decision Model and Notation, version 1.5, 2024. 2024-01. Standard. Object Management Group.
- OMG® UML: Unified Modeling Language, version 2.5, 2017. Standard. Object Management Group.

- RTCA DO-178C: Software Considerations in Airborne Systems and Equipment Certification, 2011. Standard. Radio Technical Commission for Aeronautics, RTCA Inc.

6.2 Documentos de ISTQB®

- ISTQB-ACT, ISTQB®, 2019. Certified Tester Specialist Mobile Acceptance Testing: Syllabus. Version 1.0.
- ISTQB-AI, ISTQB®, 2021. Certified Tester AI Testing: Syllabus. Version 1.0.
- ISTQB-CTFL, ISTQB®, 2024. Certified Tester Foundation Level: Syllabus. Version 4.0.1.
- ISTQB-ITP, ISTQB®, 2011. Certified Tester Expert Level Improving the Test Process: Syllabus. Version 1.0.
- ISTQB-MAT, ISTQB®, 2019. Certified Tester Specialist Mobile Application Testing: Syllabus. Version 1.0.
- ISTQB-MBT, ISTQB®, 2024. Certified Tester Model-Based Tester: Syllabus. Version 1.1.
- ISTQB-PT, ISTQB®, 2018. Certified Tester Performance Testing: Syllabus. Version 1.0.
- ISTQB-TAE, ISTQB®, 2024. Certified Tester Test Automation Engineering: Syllabus. Version 2.0.
- ISTQB-TM, ISTQB®, 2024. Certified Tester Advanced Level Test Management: Syllabus. Version 3.0.
- ISTQB-TTA, ISTQB®, 2021. Certified Tester Advanced Level Technical Test Analyst: Syllabus. Version 4.0.
- ISTQB-UT, ISTQB®, 2018. Certified Tester Usability Testing: Syllabus. Version 1.0.

6.3 Libros

- AMMANN, Paul; OFFUTT, Jeff, 2008. Introduction to Software Testing. Cambridge University Press.
- BEIZER, Boris, 1990. Software Testing Techniques (2nd Ed.) International Thomson Computer Press.
- BINDER, Robert V., 2000. Testing Object-Oriented Systems. Addison-Wesley.
- ELSAYED, Elsayed A., 2021. Reliability Engineering. Wiley.
- FORGÁCS, Istvan; KOVÁCS, Attila, 2019. Practical Test Design: Selection of traditional and automated test design techniques. BCS, The Chartered Institute for IT.

- FORGÁCS, Istvan; KOVÁCS, Attila, 2024. Modern Software Testing Techniques. Apress.
- GRADY, Robert; CASWELL, Deborah, 1987. Software Metrics: Establishing a Company-wide Program. Prentice Hall.
- HENDRICKSON, Elisabeth, 2013. Explore It! Pragmatic Bookshelf.
- JORGENSEN, Paul, 2014. Software Testing. A Craftsman's Approach. CRC Press.
- KANER, Cem; FALK, Jack; NGUYEN, Hung Q., 1999. Testing Computer Software. Wiley.
- KANER, Cem; PADMANABHAN, Sowmya; HOFFMAN, Douglas, 2013. The Domain Testing Workbook. Context Driven Press.
- KOOMEN, Tim; AALST, Leo van der; BROEKMAN, Bart; VROON, Michiel, 2006. TMap Next for result- driven testing. UTN Publishers.
- KUHN, D. Richard; YU, Lei; KACKER, Raghu N., 2013. Introduction to Combinatorial Testing. CRC Press.

6.4 Artículos

- ALYAHYA, Sultan, 2020. Crowdsourced Software Testing: A Systematic Literature Review. Information and Software Technology. Vol. 127, p. 106363. Available from doi: 10.1016/j.infsof.2020.106363.
- ANTONIOL, Giuliano; BRIAND, Lionel; DI PENTA, Massimiliano; LABICHE, Yvan, 2002. A case study using the round-trip strategy for state-based class testing. Proceedings 13th International Symposium on Software Reliability Engineering, pp. 269–279. isbn 0-7695-1763-3. Available from doi: 10.1109/ ISSRE.2002.1173268.
- ARCURI, Andrea; IQBAL, Muhammad Zohaib; BRIAND, Lionel, 2012. Random Testing: Theoretical Results and Practical Implications. IEEE Transactions on Software Engineering. Vol. 38, pp. 258–277. Available from doi: 10.1109/TSE.2011.121.
- BARR, Earl; HARMAN, Mark; MCMINN, Phil; SHAHBAZ, Muzammil; YOO, Shin, 2014. The Oracle Problem in Software Testing: A Survey. IEEE Transactions on Software Engineering. Vol. 41, no. 5, pp. 507–525. Available from doi: 10.1109/TSE.2014.2372785.
- BOCHMANN, Gregor; PETRENKO, Alexandre, 1994. Protocol Testing: Review of Methods and Relevance for Software Testing. ISSTA '94: Proceedings of the 1994 ACM SIGSOFT international symposium on Software testing and analysis, pp. 109–124. Available from doi: 10.1145/186258.187153.
- CATOLINO, Gemma; PALOMBA, Fabio; ZAIDMAN, Andy; FERRUCCI, Filomena, 2019. Not All Bugs Are the Same: Understanding, Characterizing, and Classifying Bug Types. Journal of Systems and Software. Vol. 152, pp. 165–181. Available from doi: 10.1016/j.jss.2019.03.002.

- CHILLAREGE, R. et al., 1992. Orthogonal Defect Classification - A Concept for In-Process Measurements. IEEE Transactions on Software Engineering. Vol. 18, pp. 943–956. Available from doi: 10 . 1109 / 32 . 177364.

- CHOW, Tsun, 1978. Testing Software Design Modeled by Finite-State Machines. IEEE Transactions on Software Engineering. Vol. 4, pp. 178–187. Available from doi: 10.1109/TSE.1978.231496.

- COHEN, D.M.; DALAL, Siddhartha; KAJLA, A.; PATTON, G.C., 1994. The Automatic Efficient Test Generator (AETG) system. Proceedings of 1994 IEEE International Symposium on Software Reliability Engineering, pp. 303–309. isbn 0-8186-6665-X. Available from doi: 10.1109/ISSRE.1994.341392.

- ENGSTRÖM, Emelie; RUNESON, Per; SKOGLUND, Mats, 2010. A systematic review on regression test selection techniques. Information and Software Technology. Vol. 52, pp. 14–30. Available from doi: 10.1016/j.infsof.2009.07.001.

- GHAZI, Ahmad Nauman; GARIGAPATI, Ratna; PETERSEN, Kai, 2017. Checklists to Support Test Charter Design in Exploratory Testing. Agile Processes in Software Engineering and Extreme Programming. XP 2017. isbn 978-3-319-57632-9. Available from doi: 10.1007/978-3-319-57633-6_17.

- HAREL, David, 1987. Statecharts: A Visual Formalism For Complex Systems. Science of Computer Programming. Vol. 8, pp. 231–274. Available from doi: 10.1016/0167-6423(87)90035-9.

- HUANG, Rubing; SUN, Weifeng; XU, Yinyin; CHEN, Haibo; TOWEY, Dave; XIA, Xin, 2019. A Survey on Adaptive Random Testing. IEEE Transactions on Software Engineering. Vol. 47, no. 10, pp. 2052–2083. Available from doi: 10.1109/TSE.2019.2942921.

- JENG, Bingchiang; WEYUKER, Elaine, 1994. A Simplified Domain-Testing Strategy. ACM Transactions on Software Engineering and Methodology. Vol. 3, pp. 254–270. Available from doi: 10.1145/196092. 193171.

- JUERGENS, Elmar; PAGANO, Dennis; GOEB, Andreas, 2018. Test Impact Analysis: Detecting Errors Early Despite Large, Long-Running Test Suites. Whitepaper, CQSE GmbH.

- KUHN, D.; WALLACE, Dolores; JR, A.M., 2004. Software Fault Interactions and Implications for Software Testing. IEEE Transactions on Software Engineering. Vol. 30, pp. 418–421. Available from doi: 10.1109/ TSE.2004.24.

- LEICHT, Niklas; BLOHM, Ivo; LEIMEISTER, Jan Marco, 2017. Leveraging the Power of the Crowd for Software Testing. IEEE Software. Vol. 34, pp. 62–69. Available from doi: 10.1109/MS.2017.37.

- OFFUTT, A. Jefferson, 1992. Investigations of the software testing coupling effect. ACM Transactions on Software Engineering and Methodology. Vol. 1, pp. 5–20.

- RECHTBERGER, Vaclav; BURES, Miroslav; AHMED, Bestoun, 2022. Overview of Test Coverage Criteria for Test Case Generation from Finite State Machines Modelled as Directed Graphs. IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Valencia, Spain, pp. 207–214.

- RWEMALIKA, Renaud; KINTIS, Marinos; PAPADAKIS, Mike; LE TRAON, Yves; LORRACH, Pierre, 2019. On the Evolution of Keyword-Driven Test Suites. 12th IEEE Conference on Software Testing, Validation and Verification (ICST), pp. 335–345.

- SEGURA, Sergio; FRASER, Gordon; SANCHEZ, Ana; RUIZ-CORTÉS, Antonio, 2016. A Survey on Metamorphic Testing. IEEE Transactions on Software Engineering, pp. 46–53.

- SEGURA, Sergio; TOWEY, Dave; ZHOU, Zhi Quan; CHEN, Tsong Yueh, 2020. Metamorphic Testing: Testing the Untestable. IEEE Software. Vol. 42, no. 9, pp. 805–824.

- WU, Huayao; NIE, Changhai; PETKE, Justyna; JIA, Yue; HARMAN, Mark, 2020. An Empirical Comparison of Combinatorial Testing, Random Testing and Adaptive Random Testing. IEEE Transactions on Software Engineering. Vol. 46, no. 3, pp. 302–320.

6.5 Páginas Web

1. COMMISSION, European, 2016. General Data Protection Regulation: Regulation (EU) 2016/679 [online]. [visited on 2024-09-15]. Available from: https://commission.europa.eu/law/law- topic/data-protection/ legal-framework-eu-data-protection_en?.

2. COMPUTER SOCIETY, IEEE; JTC 1/SC7, ISO/IEC, 2024. SE VOCAB: Software and Systems Engineering Vocabulary [online]. [visited on 2024-09-15]. Available from: https://pascal.computer.org/.

3. CZERWONKA, Jacek, 2004. Pairwise Testing: Combinatorial Test Case Generation [online]. [visited on 2024-09-15]. Available from: www.pairwise.org.

4. HEALTH, U.S. Department of; SERVICES, Human, 2024. Health Insurance Portability and Accountability Act: Standards for Privacy of Individually Identifiable Health Information (Privacy Rules) [online]. [visited on 2024-09-15]. Available from: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/ index.html.

5. IREB-GLOSSARY, IREB®, 2024. The CPRE Glossary: Core terms of Requirements Engineering [online]. [visited on 2024-09-15]. Available from: https://glossary.istqb.org.

6. ISTQB-GLOSSARY, ISTQB®, 2024. Standard Glossary of Terms Used in Software Testing [online]. [visited on 2024-09-15]. Available from: https://glossary.istqb.org.

7. Site of Software Test Design, 2020 [online]. [visited on 2024-09-15]. Available from: https:// test-design. org/.

8. SUMI, 1991. Software Usability Measurement Inventory: Standard evaluation questionnaire for assessing quality of use [online]. [visited on 2024-09-15]. Available from: sumi.uxp.ie.

9. U.S. DEPARTMENT OF JUSTICE, Civil Rights Division, 2010. Americans with Disabilities Act: Guidance & Resource Materials [online]. [visited on 2024-09-15]. Available from: www.ada.gov/resources/.

10. UK GOVERNMENT, Equalities Office, 2010. Equality Act 2010: guidance: Information and guidance on the Equality Act 2010, including age discrimination and public sector Equality Duty. [online]. [visited on 2024-09-15]. Available from: www.gov.uk/guidance/equality-act-2010-guidance.

11. WAMMI, 1999. Website Analysis and Measurement Inventory: Professional service for analyzing user experience [online]. [visited on 2024-09-15]. Available from: www.wammi.com.

12. WCAG, 2023. Web Content Accessibility Guidelines 2.2: W3C Recommendation [online]. [visited on 2024- 09-15]. Available from: www.w3.org/TR/WCAG22/.

Las referencias anteriores apuntan a información disponible en Internet y en otros lugares. Aunque esas referencias fueron comprobadas en el momento de la publicación de este programa de estudio, el ISTQB® no se hace responsable de la disponibilidad de dichas referencias.

7 Anexo A - Objetivos de aprendizaje/nivel cognitivo de conocimiento

Los objetivos de aprendizaje específicos de este programa de estudio figuran al principio de cada capítulo. Cada tema del programa de estudio se examinará de acuerdo con su objetivo de aprendizaje.

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: Recordar, reconocer.

Nota: Los programas de estudio del nivel avanzado no tienen objetivos de aprendizaje específicos del nivel K1. Sin embargo, se recordará el contenido del programa de estudio en los capítulos 1 - 5 y las definiciones de todos los términos que figuran como palabras clave justo debajo de los títulos de los capítulos (K1), aunque no se mencionen explícitamente en los objetivos de aprendizaje.

Nivel 2: Comprender (K2)

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

Verbos de acción: Clasificar, comparar, diferenciar, distinguir, explicar, dar ejemplos, interpretar, resumir

Ejemplos	Notas
Diferenciar entre pruebas de corrección funcional, pertinencia funcional y completitud funcional.	Buscar diferencias entre conceptos.
Explicar las pruebas CLAB (por acrónimo en inglés «CRUD»).	
Aportar ejemplos de requisitos de entorno de prueba.	
Resumir la participación del analista de prueba en varios ciclos de vida de desarrollo del software.	

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	Notas
Aplicar pruebas de dominio	Debe referirse a un procedimiento / técnica / proceso, etc.
Preparar contratos de prueba para pruebas basadas en sesiones	
Utilizar pruebas guiadas por palabra clave para desarrollar guiones de prueba	Puede utilizarse en una LO que desee que el candidato sea capaz de utilizar una técnica o procedimiento. Similar a «aplicar».

Nivel 4: Analizar (K4)

El candidato puede separar la información relacionada con un procedimiento o técnica en sus partes constituyentes para una mejor comprensión y puede distinguir entre hechos e inferencias. Una aplicación típica es analizar la situación de un documento, software o proyecto y proponer acciones adecuadas para resolver un problema o tarea.

Verbos de acción: Analizar, deconstruir, esquematizar, priorizar, seleccionar.

Ejemplos	Notas
Analizar el impacto de los cambios para determinar el alcance de las pruebas de regresión.	Evaluable sólo en combinación con un objetivo medible del análisis. Debe tener la forma «Analizar de X a Y» (o similar).
Seleccionar las técnicas de prueba adecuadas para mitigar los riesgos de producto en una situación determinada	Necesario cuando la selección necesite un análisis.

Referencia

(Para los niveles cognitivos de los objetivos de aprendizaje)

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

8 Anexo B - Matriz de trazabilidad de los resultados de negocio con respecto a los objetivos de aprendizaje

Esta sección enumera la trazabilidad entre los resultados de negocio y los objetivos de aprendizaje del analista de prueba de nivel avanzado.

Resultados de Negocio		BO - 01	BO - 02	BO - 03	BO - 04	BO - 05	BO - 06	BO - 07	BO - 08	BO - 09
TA-B01	Apoyar y llevar a cabo la prueba adecuada en función del ciclo de vida de desarrollo del software seguido	**6**								
TA-B02	Aplicar los principios de la prueba basada en el riesgo.		**3**							
TA-B03	Seleccionar y aplicar técnicas de prueba apropiadas para apoyar la consecución de los objetivos de prueba			**13**						
TA-B04	Proporcionar documentación con los niveles adecuados de detalle y calidad				**5**					
TA-B05	Determinar los tipos adecuados de pruebas funcionales que se deben realizar					**1**				
TA-B06	Contribuir a las pruebas no funcionales						**3**			
TA-B07	Contribuir a la prevención de defectos							**5**		
TA-B08	Mejorar la eficiencia del proceso de prueba con el uso de herramientas								**4**	
TA-B09	Especificar los requisitos de los entornos de prueba y los datos de prueba									**2**

Capítulo/ sección/ subsección	Objetivos de Aprendizaje	Nivel - K	BO - 01	BO - 02	BO - 03	BO - 04	BO - 05	BO - 06	BO - 07	BO - 08	BO - 09
1	**Tareas del analista de prueba en el proceso de prueba**										
1.1	**La prueba en el ciclo de vida del desarrollo de software**										
TA-1.1.1	Resumir la participación del analista de prueba en diversos ciclos de vida del desarrollo de software.	**(K2)**	X								
1.2	**Participación en actividades de prueba**										
TA-1.2.1	Resumir las tareas realizadas por el analista de prueba como parte del análisis de prueba.	**(K2)**	X								
TA-1.2.2	Resumir las tareas realizadas por el analista de prueba como parte del diseño de prueba	**(K2)**	X								
TA-1.2.3	Resumir las tareas realizadas por el analista de prueba como parte de la implementación de prueba.	**(K2)**	X								
TA-1.2.4	Resumir las tareas realizadas por el analista de prueba como parte de la ejecución de la prueba.	**(K2)**	X								
1.3	**Tareas relacionadas con productos de trabajo**										
TA-1.3.1	Diferenciar entre casos de prueba de alto nivel y casos de prueba de bajo nivel.	**(K2)**				X					

Capítulo/ sección/ subsección	Objetivos de Aprendizaje	Nivel - K	BO - 01	BO - 02	BO - 03	BO - 04	BO - 05	BO - 06	BO - 07	BO - 08	BO - 09
TA-1.3.2	Explicar los criterios de calidad para casos de prueba.	**(K2)**				X					
TA-1.3.3	Aportar ejemplos de requisitos de entorno de prueba.	**(K2)**				X					X
TA-1.3.4	Explicar el problema del oráculo de prueba y las posibles soluciones.	**(K2)**			X	X					
TA-1.3.5	Aportar ejemplos de requisitos de datos de prueba.	**(K2)**				X					X
TA-1.3.6	Utilizar pruebas guiadas por palabras clave para desarrollar guiones de prueba.	**(K3)**	X							X	
TA-1.3.7	Resumir los tipos de herramientas para gestionar los productos de prueba.	**(K2)**								X	
2	Tareas del analista de prueba en la prueba basada en el riesgo										
2.1	**Análisis del riesgo**										
TA-2.1.1	Resumir la contribución del analista de pruebas al análisis del riesgo de producto.	**(K2)**		X							
2.2	**Control del riesgo**										

Capítulo/ sección/ subsección	Objetivos de Aprendizaje	Nivel - K	BO - 01	BO - 02	BO - 03	BO - 04	BO - 05	BO - 06	BO - 07	BO - 08	BO - 09
TA-2.2.1	Analizar el impacto de cambios para determinar el alcance de la prueba de regresión.	**(K4)**		X						X	
3	**Análisis de prueba y diseño de prueba**										
3.1	**Técnicas de prueba basadas en datos**										
TA-3.1.1	Aplicar la prueba de dominio.	**(K3)**			X						
TA-3.1.2	Aplicar la prueba combinatoria.	**(K3)**			X						
TA-3.1.3	Resumir las ventajas y limitaciones de la prueba aleatoria.	**(K2)**			X						
3.2	**Técnicas de prueba basadas en el comportamiento**										
TA-3.2.1	Explicar la prueba CLAB (por acrónimo en inglés «CRUD»).	**(K2)**			X						
TA-3.2.2	Aplicar la prueba de transición de estado.	**(K3)**			X						
TA-3.2.3	Aplicar la prueba basada en escenarios.	**(K3)**			X						

Capítulo/ sección/ subsección	Objetivos de Aprendizaje	Nivel - K	BO - 01	BO - 02	BO - 03	BO - 04	BO - 05	BO - 06	BO - 07	BO - 08	BO - 09
3.3	**Técnicas de prueba basadas en reglas**										
TA-3.3.1	Aplicar la prueba de tabla de decisión.	**(K3)**			X						
TA-3.3.2	Aplicar la prueba metamórfica.	**(K3)**			X						
3.4	Prueba basada en la experiencia										
TA-3.4.1	Preparar contratos de prueba para la prueba basada en sesiones.	**(K3)**			X						
TA-3.4.2	Preparar listas de comprobación que apoyen la prueba basada en la experiencia.	**(K3)**			X						
TA-3.4.3	Aportar ejemplos de las ventajas y limitaciones de la prueba multitudinaria.	**(K2)**			X						
3.5	**Aplicación de las técnicas de prueba más adecuadas**										
TA-3.5.1	Seleccionar las técnicas de prueba adecuadas para mitigar riesgos de producto en una situación determinada.	**(K4)**		X	X						
TA-3.5.2	Explicar las ventajas y los riesgos de automatizar el diseño de prueba.	**(K2)**								X	

Capítulo/ sección/ subsección	Objetivos de Aprendizaje	Nivel - K	BO - 01	BO - 02	BO - 03	BO - 04	BO - 05	BO - 06	BO - 07	BO - 08	BO - 09
4	**Prueba de características de calidad**										
4.1	**Prueba funcional**										
TA-4.1.1	Diferenciar entre prueba de corrección funcional, pertinencia funcional y completitud funcional.	**(K2)**					X				
4.2	**Prueba de usabilidad**										
TA-4.2.1	Explicar cómo contribuye el analista de prueba a la prueba de usabilidad.	**(K2)**						X			
4.3	**Prueba de flexibilidad**										
TA-4.3.1	Explicar cómo contribuye el analista de prueba a la prueba de adaptabilidad e instalabilidad.	**(K2)**						X			
4.4	**Prueba de compatibilidad**										
TA-4.4.1	Explicar cómo contribuye el analista de prueba a la prueba de interoperabilidad.	**(K2)**						X			
5	**Prevención de defectos de software**										

Capítulo/ sección/ subsección	Objetivos de Aprendizaje	Nivel - K	BO - 01	BO - 02	BO - 03	BO - 04	BO - 05	BO - 06	BO - 07	BO - 08	BO - 09
	5.1 Prácticas de prevención de defectos										
TA-5.2.1	Explicar cómo el analista de prueba puede contribuir a la prevención de defectos.	**(K2)**							X		
	5.2 Apoyo a la contención de fase										
TA-5.2.1	Utilizar un modelo del objeto de prueba para detectar defectos en una especificación.	**(K3)**							X		
TA-5.2.2	Aplicar una técnica de revisión a una base de prueba para encontrar defectos.	**(K3)**							X		
	5.2 Mitigar la recurrencia de defectos										
TA-5.3.1	Analizar los resultados de pruebas para identificar posibles mejoras en la detección de defectos.	**(K4)**							X		
TA-5.3.2	Explicar cómo la clasificación de defectos apoya el análisis de la causa raíz.	**(K2)**							X		

9 Anexo C - Notas de la versión

Programa de estudio de Analista de Pruebas de Nivel Avanzado de ISTQB® v4.0 es una actualización mayor. Por esta razón, no hay notas de la versión detalladas por capítulo y sección. Sin embargo, a continuación, se proporciona un resumen de los principales cambios. Además, en un documento separado de Notas de la versión, ISTQB® proporciona una trazabilidad detallada entre los objetivos de aprendizaje de las versiones v3.1.2 y v4.0 del programa de estudios de Analista de Pruebas de Nivel Avanzado.

En esta versión principal se han introducido los siguientes cambios

Estructura del programa de estudio

Se han editado todos los objetivos de aprendizaje para hacerlos atómicos y crear una trazabilidad uno a uno de los objetivos de aprendizaje al contenido para evitar tener contenido sin un objetivo de aprendizaje correspondiente. Cada objetivo de aprendizaje se describe en una sección con el mismo número (por ejemplo, TA-1.1.1 se trata en la sección 1.1.1). El objetivo era que esta versión fuera más fácil de leer, comprender, aprender y traducir, centrándose en aumentar la utilidad práctica y el equilibrio entre conocimientos y habilidades.

En la v4.0 hay 36 objetivos de aprendizaje, frente a los 31 de la v3.1.2. Varios objetivos de aprendizaje K4 se redujeron a K2 o K3. En el caso de los objetivos de aprendizaje relacionados con las técnicas de prueba, la motivación fue que la atención debía centrarse en la aplicación de las técnicas de prueba y no en el análisis de la documentación para el posterior diseño de las pruebas. Algunos objetivos de aprendizaje se fusionaron en un único objetivo de aprendizaje. La tabla siguiente muestra estadísticas detalladas sobre el número de objetivos de aprendizaje y el tiempo de formación.

Versión del programa de estudio	#K2 OA (tiempo)	#K3 OA (tiempo)	#K4 OA (tiempo)	#Total OA (tiempo)
actual (4.0)	22 (330 min)	11 (660 min)	3 (225 min)	36 (1215 min)
anterior (3.1.2)	16 (240 min)	5 (300 min)	10 (750 min)	31 (1290 min)

Dónde

- OA: Objetivo de Aprendizaje

Clasificación de las técnicas de prueba

La estructura del capítulo 3 refleja una clasificación más detallada de las técnicas de prueba en comparación con la versión 3.1.2. Las técnicas de prueba de caja negra se clasifican ahora en basadas en datos, basadas en comportamientos y basadas en reglas, según el tipo de modelo de objetos de prueba subyacente.

Ampliación del alcance del capítulo 5

El antiguo capítulo 5 (dedicado exclusivamente a las revisiones) se amplió para tratar otras formas esenciales de prácticas de prevención de defectos utilizadas por el AP, como el uso de modelos para detectar defectos en las especificaciones, el análisis de los resultados de las pruebas para mejorar la detección de defectos y el uso de la clasificación de defectos para apoyar el análisis de la causa raíz.

Cambios en los objetivos de aprendizaje

- El capítulo 1 se reorganizó en una sección sobre las actividades de prueba y otra sobre el producto de prueba.
- El tema sobre casos de prueba de alto nivel y casos de prueba de bajo nivel (antiguo TA-1.4.2) pasó de K4 a K2 (TA-1.3.1).
- El OA K3 sobre pruebas basadas en el riesgo (antiguo TA-2.1.1) se dividió en dos, el K2 TA-2.1.1 relativo al análisis de riesgos y el K4 TA-2.2.1 relativo al control de riesgos.
- La partición de equivalencias (antigua K4 TA-3.2.1) y el análisis del valor frontera (antigua K4 TA-3.2.2) se sustituyeron por la más general K3 TA-3.1.1 sobre pruebas de dominio.
- La prueba de a pares (antiguo K4 TA-3.2.6) y los diagramas de árbol de clasificación (antiguo K2 TA-3.2.5) se fusionaron en un único y más general K3 TA-3.1.2 sobre pruebas combinatorias.
- La prueba de transición de estados (antiguo K4 TA-3.2.4) se sustituyó por el K3 TA-3.2.2, que se centra en la cobertura N-conmutaciones y de ida y vuelta. Se han eliminado las redundancias con CTFL.
- Las pruebas de casos de uso (antiguo K4 TA-3.2.7) se sustituyeron por el más general K3 TA-3.2.3 sobre pruebas basadas en escenarios.
- La prueba de tablas de decisión (antiguo K4 TA-3.2.3) se rebajó a K3 (TA-3.3.1).
- La prueba exploratoria (antiguo K3 TA-3.3.2) se sustituyó por dos OA independientes: sobre la preparación de cartas de prueba (K3 TA-3.4.1) y sobre la preparación de listas de comprobación que apoyen la prueba basada en la experiencia (K3 TA-3.4.2). Se han eliminado las redundancias con la actual CTFL v4.0.
- Cuatro OA relacionados con la comparación de técnicas de prueba y la aplicación de la técnica más apropiada (antiguo: K4 TA-3.2.8, K2 TA-3.3.3, K2 TA-3.4.1, K2 TA-3.3.1) se combinaron en un K4 TA-3.5.1.
- Cuatro OA sobre pruebas funcionales (antiguos: K2 TA-4.2.1, K2 TA-4.2.2, K2 TA-4.2.3 y K4 TA-4.2.7) se combinaron en un K2 TA-4.1.1. El tema se simplificó porque las pruebas funcionales ya se describen en gran medida en el contexto de las técnicas de prueba en el capítulo 3.
- Los temas del capítulo 6 (Herramientas de prueba) se trasladaron a la sección 1.3, centrándose en cuestiones más prácticas. El antiguo K3 TA-6.2.1 «Para un escenario dado, determine las actividades apropiadas para un analista de prueba en un proyecto de pruebas impulsadas por palabras clave» se cambió ligeramente a K3 TA-1.3.6 «Utilizar la prueba impulsada por palabras clave para desarrollar

guiones de prueba». El antiguo K2 TA-6.3.1 'Explicar el uso y los tipos de herramientas de prueba aplicadas en el diseño de pruebas, la preparación de datos de prueba y la ejecución de pruebas' se cambió ligeramente por el K2 TA-1.3.7 'Resumir los tipos de herramientas aplicadas en la gestión de productos de prueba'.

- Dos OA similares sobre revisiones (antiguas K3 TA-5.2.1 y K3 TA-5.2.2) se combinaron en un solo K3 TA-5.2.2.

Temas nuevos

- Criterios de calidad para casos de prueba (K2 TA-1.3.2)
- Requisitos del entorno de prueba (K2 TA-1.3.3)
- Determinación de los oráculos de prueba (K2 TA-1.3.4)
- Requisitos de los datos de prueba (K2 TA-1.3.5)
- Pruebas aleatorias (K2 TA-3.1.3)
- Prueba CLAB (K2 TA-3.2.1)
- Pruebas metamórficas (K3 TA-3.3.2)
- Pruebas multitudinarias (K2 TA-3.4.3)
- Ventajas y riesgos de automatizar el diseño de las pruebas (K2 TA-3.5.2)
- Contribuciones del analista de pruebas a la prevención de defectos (K2 TA-5.1.1)
- Uso de modelos para detectar defectos en las especificaciones (K3 TA-5.2.1)
- Analizar los resultados de las pruebas para mejorar la detección de defectos (K4 TA-5.3.1)
- Apoyo al análisis de la causa raíz con la clasificación de defectos (K2 TA-5.3.2)

Actualizaciones de estándares

Los cambios en las últimas versiones de las normas internacionales para la calidad del software (ISO/IEC 25010, 2023) y las técnicas de prueba (ISO/IEC/IEEE 29119-4, 2021) han dado lugar a la adaptación del contenido correspondiente del programa de estudios.

10 Anexo D - Lista de acrónimos y abreviaturas

Descripción en inglés	Término en inglés	Término en español	Descripción en español
artificial intelligence	AI	IA	inteligencia artificial
Business Process Model and Notation	BPMN	BPMN	Business Process Model and Notation
boundary value analysis	BVA	AVF	análisis de valores frontera
create, read, update, and delete	CRUD	CLAB	crear, leer, actualizar, borrar
comma-separated value	CSV	VSC	valor separado por coma
defect detection percentage	DDP	PDD	porcentaje de detección de defectos
defect removal efficiency	DRE	EED	eficiencia de eliminación de defectos
equivalence partitioning	EP	PE	partición de equivalencia
General Data Protection Regulation	GDPR	GDPR	general data protection regulation
JavaScript Object Notation	JSON	JSON	javascript object notation
model-based testing	MBT	PBM	prueba basada en modelo
metamorphic relation	MR	RM	relación metamórfica
metamorphic testing	MT	PM	prueba metamórfica
orthogonal defect classification	ODC	COD	clasificación ortogonal de defectos

Descripción en inglés	Término en inglés	Término en español	Descripción en español
phase containment effectiveness	**PCE**	**ECF**	**efectividad de contención de fase**
root cause analysis	**RCA**	**ACR**	**Análisis de la causa raíz**
software development lifecycle	**SDLC**	**CVDS**	**ciclo de vida del desarrollo de software**
test analyst	**TA**	**AP**	**Analista de prueba**
technical test analyst	**TTA**	**APT**	**Analista de prueba técnica**
Unified Modeling Language	**UML**	**UML**	**Unified Modeling Language**
user experience	**UX**		**Experiencia de usuario**
Extensible Markup Language	**XML**	**XML**	**Extensible Markup Language**

11 Anexo E - Términos específicos de dominio

Acrónimos y Abreviaturas			
Término (idioma fuente)	**Término (español)**	**Descripción**	
CRUD	**CLAB**	**ES**	crear, leer, actualizar, borrar
		EN	create, read, update, delete
CRUD matrix	**matriz CLAB**	**ES**	Matriz que indica los tipos de acción de las funciones sobre las entidades de un sistema.
		EN	A matrix that indicates the action types of the functions on the entities within a system.
data semantics	**semántica de datos**	**ES**	El significado y la interpretación de los datos.
		EN	The meaning and interpretation of data.

Acrónimos y Abreviaturas			
Término (idioma fuente)	**Término (español)**	**Descripción**	
five whys technique	**técnica de los cinco porqués**	**ES**	Técnica interrogativa iterativa utilizada para determinar la causa raíz de un defecto o problema repitiendo la pregunta «¿por qué?» cinco veces, dirigiendo cada vez el «por qué» actual a la respuesta del «por qué» anterior.
		EN	An iterative interrogative technique used to determine the root cause of a defect or problem by repeating the question 'why?' five times, each time directing the current 'why' to the answer of the previous 'why'.
Pareto analysis	**Análisis de Pareto**	**ES**	Un enfoque para identificar las áreas problemáticas o las tareas que tendrán la mayor rentabilidad.
		EN	An approach to identify problem areas or tasks that will have the biggest payoff.
user journey map	**mapa del recorrido de usuario**	**ES**	Documentación de visualización de la experiencia del usuario que muestra los pasos que sigue un usuario en un proceso para lograr un objetivo.
		EN	User experience visualization documentation that shows the steps that a user takes in a process to accomplish a goal.
user research	**investigación de usuario**	**ES**	Disciplina que consiste en conocer las necesidades y los procesos de pensamiento de los usuarios estudiando cómo realizan las tareas, observando cómo interactúan con un componente o sistema, o mediante el análisis y la interpretación de datos.
		EN	A discipline of learning about users' needs and thought processes by studying how they perform tasks, observing how they interact with a component or system, or by data analysis and interpretation.

Tabla 1 – Acrónimos y Abreviaturas

- **Dónde:**
- **Columna: Idioma del tipo de término**
  - **EN: inglés**
  - **ES: español**

12 Anexo F - Modelo de calidad del software

La siguiente tabla muestra las características de calidad del modelo de calidad de producto ISO/IEC 25010 (ISO/IEC 25010, 2023). Indica qué características/subcaracterísticas se tratan dentro de este syllabus (TA) y cuáles se cubren en otros syllabi ISTQB® (Analista de Prueba Técnica (TTA), Pruebas de Rendimiento (PT), Prueba de Usabilidad (UT), Probador de Software Automotriz (AuT), y Probador de Seguridad (SEC)). Si varios programas de estudios cubren una característica de calidad, se enumera en primer lugar el que la cubre con más detalle. La tabla también compara el modelo ISO/IEC 25010 actual con la versión de 2011 utilizada en la versión anterior de este programa de estudios.

ISO/IEC 25010:2023 (vigente)		ISO/IEC 25010:2011 (anterior)		Notas	Programas de estudio ISTQB
inglés	español	inglés	español	español	español
Functional suitability	**Adecuación funcional**	**Functional suitability**	**Adecuación funcional**		TA
Functional completeness	Completitud funcional	Functional completeness	Completitud funcional		
Functional correctness	Corrección funcional	Functional correctness	Corrección funcional		
Functional appropriateness	Pertinencia funcional	Functional appropriateness	Pertinencia funcional		
Performance efficiency	**Eficiencia de desempeño**	**Performance efficiency**	**Eficiencia de desempeño**		PT, TTA
Time behavior	Comportamiento temporal	Time behavior	Comportamiento temporal		
Resource utilization	Utilización de recursos	Resource utilization	Utilización de recursos		
Capacity	Capacidad	Capacity	Capacidad		

ISO/IEC 25010:2023 (vigente)		ISO/IEC 25010:2011 (anterior)		Notas	Programas de estudio ISTQB
inglés	español	inglés	español	español	español
Compatibility	**Compatibilidad**	**Compatibility**	**Compatibilidad**		TA, TTA
Co-existence	Coexistencia	Co-existence	Coexistencia		TTA
Interoperability	Interoperabilidad	Interoperability	Interoperabilidad		TA
Interaction capability	**Usabilidad**	**Usability**	**Usabilidad**	Renombrado	UT, TA
Appropriateness recognizability	Recognoscibilidad de pertinencia	Appropriateness recognizability	Recognoscibilidad de pertinencia		
Learnability	Capacidad de ser aprendido	Learnability	Capacidad de ser aprendido		
Operability	Operabilidad	Operability	Operabilidad		
User error protection	Protección contra errores del usuario	User error protection	Protección contra errores del usuario		
User engagement	Compromiso de usuario	User interface aesthetics		Renombrado	
Inclusivity	Inclusividad	Accessibility	Accesibilidad	Split and renamed	
User assistance	Asistencia al usuario				
Self-descriptiveness	Capacidad de autodescripción			Nuevo	

ISO/IEC 25010:2023 (vigente)		ISO/IEC 25010:2011 (anterior)		Notas	Programas de estudio ISTQB
inglés	español	inglés	español	español	español
Reliability	**Fiabilidad**	**Reliability**	**Fiabilidad**		TTA
Faultlessness	Ausencia de defectos	Maturity	Madurez	Renombrado	
Availability	Disponibilidad	Availability	Disponibilidad		
Fault tolerance	Tolerancia a defectos	Fault tolerance	Tolerancia a defectos		
Recoverability	Recuperabilidad	Recoverability	Recuperabilidad		
Security	**Seguridad**	**Security**	**Seguridad**		SEC, TTA
Confidentiality	Confidencialidad	Confidentiality	Confidencialidad		
Integrity	Integridad	Integrity	Integridad		
Non-repudiation	No repudio	Non-repudiation	No repudio		
Accountability	Responsabilidad	Accountability	Responsabilidad		
Authenticity	Autenticidad	Authenticity	Autenticidad		
Resistance	Resistencia			Nuevo	

ISO/IEC 25010:2023 (vigente)		ISO/IEC 25010:2011 (anterior)		Notas	Programas de estudio ISTQB
inglés	español	inglés	español	español	español
Maintainability	**Mantenibilidad**	**Maintainability**	**Mantenibilidad**		TTA
Modularity	Modularidad	Modularity	Modularidad		
Reusability	Capacidad de reutilización	Reusability	Capacidad de reutilización		
Analysability	Analizabilidad	Analysability	Analizabilidad		
Modifiability	Capacidad de ser modificado	Modifiability	Capacidad de ser modificado		
Testability	Capacidad de ser probado	Testability	Capacidad de ser probado		
Flexibility	**Flexibilidad**	**Portability**	**Portabilidad**	Renombrado	TTA, TA, PT
Adaptability	Adaptabilidad	Adaptability	Adaptabilidad		TA, TTA
Scalability	Escalabilidad			Nuevo	PT
Installability	Instalabilidad	Installability	Instalabilidad		TA, TTA
Replaceability	Capacidad de ser reemplazado	Replaceability	Capacidad de ser reemplazado		TTA

ISO/IEC 25010:2023 (vigente)		ISO/IEC 25010:2011 (anterior)		Notas	Programas de estudio ISTQB
inglés	español	inglés	español	español	español
Safety	**Protección**			Nuevo	AuT
Operational constraint	Restricción operativa			Nuevo	
Risk identification	Identificación de riesgos			Nuevo	
Fail safe	Protección ante fallos			Nuevo	
Hazard warning	Advertencia de peligro			Nuevo	
Safe integration	Integración protegida			Nuevo	

13 Anexo G - Marcas registradas

ISTQB® es una marca registrada de International Software Testing Qualifications Board.

UML® es una marca registrada de Object Management Group (OMG).

BPMN™ es una marca registrada de Object Management Group (OMG).