Principios de las Pruebas de Software

Formación para el Probador Certificado Nivel Básico del ISTQB

"ISTQB Certified Tester - Foundation Level"

Programa de estudios versión 4.0 (2023)

mtp DIGITAL BUSINESS ASSURANCE

www.mtp.es

www.mtp.es
C/ Santa Leonor, 65 • Edif. C • PL. 4ª • 28037 Madrid (Spain)
Tel.: +34 91 144 06 00 • Fax: +34 91 304 49 15

mtp DIGITAL BUSINESS ASSURANCE

Principios de las Pruebas de Software

Formación para el Probador Certificado del ISTQB® - Nivel básico

Correspondiente al programa de estudios versión 4 – Doc. v2.1

Material de formación ISTQB Nivel Básico versión v4 (2023)
acreditado por el SSTQB / ISTQB el 28 de junio de 2024

Versión del documento acreditada v2.1

itegGO

C.I.F. B-88056163

Material propiedad de MTP

Propiedad intelectual y derechos de autor

- La propiedad intelectual y los derechos de autor sobre los productos y servicios pertenecen exclusivamente a MTP. EL CLIENTE no podrá utilizar los productos, documentación entregada y servicios recibidos sino exclusivamente para los fines de capacitación de los asistentes a la formación objeto de este curso. El destinatario de este servicio o proyecto únicamente tiene derecho a un uso privado de los documentos contenidos en el mismo, y necesita autorización expresa de MTP para modificarlos, reproducirlos, explotarlos y especialmente comercializarlos, o hacer uso de cualquier derecho perteneciente a su titular.

Capítulo 0 – Introducción

- 0/01 Objetivos
- 0/02 Contenido
- 0/03 Programa
- 0/04 Documentación
- 0/05 Evaluación
- 0/06 Niveles de Conocimiento
- 0/07 Destinatarios
- 0/08 ISTQB
- 0/09 Probador certificado

0 – Introducción

01 – Destinatarios

La certificación de Nivel Básico está dirigida a **cualquier persona** que se encuentre involucrada en la prueba de software.

Incluidas las personas que asumen los roles de

- probadores,
- analistas de prueba,
- ingenieros de prueba,
- consultores de prueba,
- jefes de prueba,
- probadores que participan en la prueba de aceptación y
- desarrolladores de software.

0 – Introducción

01 – Destinatarios

Esta cualificación de nivel básico también es apropiada para cualquier persona que desee una **comprensión básica** de la prueba de software, como:

- jefes de proyecto,
- responsables de calidad,
- propietarios de producto,
- jefes de desarrollo de software,
- analistas de negocio,
- directores de TI y
- consultores de gestión.

Los titulares del Certificado Básico podrán acceder a las certificaciones de nivel superior en la prueba de software.

0 – Introducción

02 – Desarrollo profesional

El esquema ISTQB® proporciona apoyo a los profesionales de la prueba en todas las etapas de su carrera ofreciendo tanto amplitud como profundidad de conocimientos.

Las personas que hayan obtenido la certificación de la Fundación ISTQB® también pueden estar interesadas en el **nivel avanzado troncal**

- analista de prueba,
- analista de pruebas técnicas y
- jefe de prueba

A partir de la certificación como jefe de prueba se puede obtener el nivel experto

- Gestión de la Prueba o
- Mejora del Proceso de Prueba

Cualquiera persona que busque desarrollar competencias en prácticas de **prueba en un entorno Ágil** podría considerar las certificaciones de Probador Técnico Ágil o Liderazgo de Prueba Ágil a Escala.

0 – Introducción

02 – Desarrollo profesional

La corriente de **Especialista** ofrece una inmersión profunda en áreas que:

- tienen **enfoques de prueba específicos y actividades de prueba**. Por ejemplo, en
  - automatización de la prueba,
  - prueba de IA,
  - prueba basada en modelos,
  - prueba de aplicaciones móviles
- que están relacionadas con **áreas de prueba específicas**. Por ejemplo:
  - prueba de rendimiento,
  - prueba de usabilidad,
  - prueba de aceptación,
  - prueba de seguridad
- o que agrupan conocimientos de **prueba para ciertos dominios de la industria**. Por ejemplo:
  - automoción o
  - juegos

Visite www.istqb.org para obtener la información más reciente sobre el programa de probadores certificados de ISTQB.

0 – Introducción

03 – Objetivos

Formación "Principios de las pruebas de Software"

- Proporciona al participante conocimientos básicos en el campo de las Pruebas de Software.
- Constituye la base de evaluación para el "Probador Certificado – Nivel Básico".

Los objetivos de la formación para el Probador Certificado son, entre otros:

- Proporcionar conocimientos acerca de las pruebas de software sobre una base fundamentada (marco estándar)
- Poder concebir las pruebas de software específicamente para un proyecto
- Poder elegir de forma adecuada las técnicas para las pruebas y sus objetivos
- Poder escoger y utilizar de manera adecuada las herramientas de apoyo para las pruebas.

0 – Introducción

03 – Objetivos. Resultados de negocio esperados (1/2)

Un probador certificado en el nivel básico puede:

- FL - BO - 01 Comprender qué es probar y por qué es beneficioso
- FL - BO - 02 Comprender los conceptos fundamentales de la prueba de software
- FL - BO - 03 Identificar el enfoque de prueba y las actividades que se deben implementar en función del contexto de prueba
- FL - BO - 04 Evaluar y mejorar la calidad de la documentación
- FL - BO - 05 Aumentar la efectividad y eficiencia de la prueba
- FL - BO - 06 Alinear el proceso de prueba con el ciclo de vida de desarrollo de software
- FL - BO - 07 Comprender los principios de gestión de la prueba

0 – Introducción

03 – Objetivos. Resultados de negocio esperados (2/2)

Un probador certificado en el nivel básico puede:

- FL - BO - 08 Redactar y comunicar informes de defectos claros y comprensibles
- FL - BO - 09 Comprender los factores que influyen en las prioridades y esfuerzos relacionados con la prueba
- FL - BO - 10 Trabajar como parte de un equipo interfuncional
- FL - BO - 11 Conocer los riesgos y beneficios relacionados con la automatización de la prueba
- FL - BO - 12 Identificar las competencias esenciales necesarias para probar
- FL - BO - 13 Comprender el impacto del riesgo en las pruebas
- FL - BO - 14 Informar eficazmente sobre el avance y la calidad de la prueba

0 – Introducción

04 – Objetivos de aprendizaje

En cada sección del Programa de estudio se indican los objetivos de aprendizaje correspondientes, y se clasifican según lo siguiente (consultar Apéndice correspondiente - Objetivos de Aprendizaje/Nivel Cognitivo de Conocimiento del programa de estudio)

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

- Verbos de acción: identificar, recordar, rememorar, reconocer.
- Ejemplos:
  - "Identificar los objetivos característicos de la prueba".
  - "Recordar los conceptos de la pirámide de prueba".
  - "Reconocer cómo un probador añade valor a la planificación de la iteración y la entrega".

0 – Introducción

04 – Objetivos de aprendizaje

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

- Verbos de acción: clasificar, comparar, contrastar, diferenciar, distinguir, ejemplificar, explicar, dar ejemplos, interpretar, resumir.
- Ejemplos:
  - "Clasificar las diferentes opciones para redactar los criterios de aceptación".
  - "Comparar los diferentes roles en la prueba" (buscar similitudes, diferencias o ambas).
  - "Distinguir entre riesgos de proyecto y riesgos de producto" (permite diferenciar conceptos).
  - "Dar ejemplos de la finalidad y el contenido de un plan de prueba".
  - "Explicar el impacto del contexto en el proceso de prueba".
  - "Resumir las actividades del proceso de revisión".

0 – Introducción

04 – Objetivos de aprendizaje

- **Nivel 3: Aplicar (K3)**: el candidato puede llevar a cabo un procedimiento cuando se enfrenta a una tarea conocida, o seleccionar el procedimiento correcto y aplicarlo a un contexto determinado.
  - Verbos de acción: aplicar, implementar, preparar, utilizar.
  - Ejemplos:
    - "Aplicar la priorización de casos de prueba" (debe referirse a un procedimiento, técnica, proceso, algoritmo, etc.).
    - "Preparar un informe de defectos".
    - "Utilizar el análisis del valor frontera para obtener casos de prueba".

0 – Introducción

04 – Objetivos de aprendizaje

- Nivel 4: K4: saber analizar y proponer acciones apropiadas para solucionar un problema o abordar una tarea. **No existen objetivos de aprendizaje de este tipo en este programa de estudios**

NOTA: cada tema se examinará según sus objetivos de aprendizaje (K).

0 – Introducción

05 – El examen de certificación

El manual se basa en los contenidos del Programa de Estudio de Nivel Básico del Probador Certificado del ISTQB® - Versión 4 (Mayo de 2023):

- Tras la finalización del seminario se podrá llevar a cabo la evaluación para el Nivel básico para el probador certificado (que no caduca).
- El examen se llevará a cabo por un examinador independiente.
- El examen que estará dividido en diferentes áreas, de acuerdo a los capítulos de este seminario, se compone de preguntas de elección múltiple.
- El examen será en **español** con una duración de **60 minutos**.
- **Es necesario obtener un 65% (26 de 40)**

0 – Introducción

05 – El examen de certificación

El manual se basa en los contenidos del Programa de Estudio de Nivel Básico del Probador Certificado del ISTQB® - Versión 4 (Mayo de 2023):

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

0 – Introducción

06 – Contenido

El seminario está dividido en siete capítulos de acuerdo con el plan de formación del ISTQB:

Capítulo **0** Introducción
Capítulo **I** Fundamentos del Proceso de Prueba
Capítulo **II** Prueba a lo Largo del Ciclo de Vida de Desarrollo de Software
Capítulo **III** Prueba Estática
Capítulo **IV** Análisis y Diseño de la Prueba
Capítulo **V** Gestión de la Prueba
Capítulo **VI** Soporte de Herramientas para el Proceso de Prueba

0 – Introducción

06 – Contenido

Hay seis capítulos con contenido que puede ser objeto de examen.

Para los cursos de formación acreditados, el programa de estudio exige un mínimo de 1135 minutos (18 horas y 55 minutos) de formación, distribuidos en los seis capítulos de la siguiente manera:

- **Capítulo 1: Fundamentos del Proceso de Prueba** (180 minutos)
  - El alumno aprende los principios básicos relacionados con la prueba, las razones por las que es necesario probar y cuáles son los objetivos de la prueba.
  - El alumno comprende el proceso de prueba, las principales actividades de prueba y el producto de prueba.
  - El alumno comprende las competencias esenciales para probar.

0 – Introducción

06 – Contenido

- **Capítulo 2: Prueba a lo Largo del Ciclo de Vida de Desarrollo de Software** (130 minutos)
  - El alumno aprende cómo se incorpora la prueba a los diferentes enfoques de desarrollo.
  - El alumno aprende los conceptos de los enfoques de probar primero, así como DevOps.
  - El alumno aprende sobre los diferentes niveles de prueba, tipos de prueba y prueba de mantenimiento.
- **Capítulo 3: Prueba Estática** (80 minutos)
  - El alumno aprende los fundamentos de la prueba estática, el proceso de retroalimentación y revisión.
- **Capítulo 4: Análisis y Diseño de la Prueba** (390 minutos)
  - El alumno aprende a aplicar técnicas de caja negra, caja blanca y prueba basada en la experiencia para obtener casos de prueba a partir de diversos productos software.
  - El alumno aprende sobre el enfoque de prueba basado en la colaboración.

0 – Introducción

06 – Contenido

- **Capítulo 5: Gestión de las Actividades de Prueba** (335 minutos)
  - El alumno aprende a planificar las pruebas en general y a estimar el esfuerzo de la prueba.
  - El alumno aprende cómo los riesgos pueden influir en el alcance de la prueba.
  - El alumno aprende a monitorizar y controlar las actividades de prueba.
  - El alumno aprende cómo la gestión de la configuración apoya la prueba.
  - El alumno aprende a informar de los defectos de forma clara y comprensible.
- **Capítulo 6: Herramientas de Prueba** (20 minutos)
  - El alumno aprende a clasificar las herramientas y a comprender los riesgos y las ventajas de la automatización de la prueba.

0 – Introducción

07 – Organización

La formación, con posible evaluación final por parte del ISTQB, incluye :

- Preparación teórica del material de estudio y una serie de ejemplos para determinados comportamientos técnicos.
- Aclaración de dudas.
- Ejercicios en profundidad acerca del material de estudio.
- Discusión sobre los temas aprendidos.

0 – Introducción

08 – Documentación

- **El material de apoyo se compone de:**
  - Un manual completo (este documento).
  - Ejercicios de aplicación de los contenidos.
  - Un conjunto de preguntas de tipo test para preparar el examen.
  - Listas de bibliografía, estándares, etc.
- **Referencias:**
  - Programa de Estudio de Nivel Básico del Probador Certificado del ISTQB® - Versión 4
  - Glosario estándar de términos utilizados en pruebas software versión en Español
- **Se recomienda la lectura del Programa de Estudio y la consulta del Glosario**

0 – Introducción

09 – ISTQB / Organizaciones Internacionales

Programa de cualificación del ISTQB

- En 1998 se definió en Gran Bretaña un programa de cualificación multi-etapa (ISEB). Historia y fechas relevantes: https://www.istqb.org/about-us/history.html
- ISTQB se fundó en 2002 como entidad sin ánimo de lucro https://www.istqb.org/about-us.html
- Existen comités de pruebas específicos para cada país que forman conjuntamente el ISTQB - International Software Testing Qualification Board. En España existe el Spanish Software Testing Qualification Board (https://es.sstqb.com/)
- Los comités de pruebas específicos de cada país son responsables de la acreditación de los proveedores de formación y de la realización de exámenes en los respectivos países.

0 – Introducción

10 – Probador certificado

Programa de cualificación del iSTQB

- Como instancia independiente, el comité verifica los cursos ofrecidos en base a los criterios definidos y otorga la acreditación al proveedor de formación.
- A pesar de que la comprobación y las pruebas de software tienen en la práctica una gran importancia debido a la irrupción de las tecnologías de la información en prácticamente todos los ámbitos de la vida, existen pocos cursos de aprendizaje y seminarios que se ocupen de manera intensiva de esta problemática.
- Esto ha llevado al ISTQB a reavivar las ya mencionadas medidas de cualificación en diferentes etapas.

0 – Introducción

10 – Probador certificado

Queremos dirigirnos a probadores de software en empresas de software y empresas industriales que quieran asentar sus conocimientos sobre unas bases fundamentadas.

¡El aprendizaje durante toda la vida es imprescindible, especialmente en el sector de las TI!

Capítulo I – Fundamentos del Proceso de Prueba

- I/01 ¿Qué es Probar?
- I/02 ¿Por qué es necesario probar?.
- I/03 Principios de la Prueba
- I/04 Actividades de Prueba, Productos de Prueba y Roles de Prueba
- I/05 Competencias Esenciales y Buenas Prácticas en la Prueba

I – Fundamentos del Proceso de Prueba

Glosario / Palabras clave en Español e Inglés

- Cobertura - coverage
- Depurar - debugging
- Defecto - defect
- Error - error
- Fallo - failure
- Calidad - quality
- Aseguramiento de la calidad - quality assurance
- Causa raíz - root cause
- Análisis de prueba - test analysis
- Base de prueba - test basis
- Caso de prueba - test case
- Compleción de la prueba - test completion
- Condición de prueba - test condition
- Control de la prueba - test control
- Datos de prueba - test data

I – Fundamentos del Proceso de Prueba

Glosario / Palabras clave en Español e Inglés

- Diseño de la prueba - test design
- Ejecución de prueba - test execution
- Implementación de prueba - test implementation
- Monitorización de la prueba - test monitoring
- Objeto de prueba - test object
- Objetivo de prueba - test objective
- Planificación de prueba - test planning
- Procedimiento de prueba - test procedure
- Resultado de prueba - test result
- Prueba - testing
- Producto de prueba – testware
- Validación - validation
- Verificación – verification

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

I – Fundamentos del Proceso de Prueba

Objetivos de Aprendizaje

1.1 ¿Qué es Probar?

- FL-1.1.1 (K1) Identificar objetivos de prueba característicos.
- FL-1.1.2 (K2) Diferenciar entre probar y depurar.

1.2 ¿Por qué es Necesario Probar?

- FL-1.2.1 (K2) Aportar ejemplos de por qué es necesario realizar pruebas.
- FL-1.2.2 (K1) Recordar la relación entre prueba y aseguramiento de la calidad.
- FL-1.2.3 (K2) Distinguir entre causa raíz, error, defecto y fallo.

1.3 Principios de la Prueba

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

I – Fundamentos del Proceso de Prueba

Objetivos de Aprendizaje

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

- FL-1.4.1 (K2) Resumir las diferentes actividades y tareas de prueba.
- FL-1.4.2 (K2) Explicar el impacto del contexto en el proceso de prueba.
- FL-1.4.3 (K2) Diferenciar el producto de prueba que soporta las actividades de prueba.
- FL-1.4.4 (K2) Explicar el valor de mantener la trazabilidad.
- FL-1.4.5 (K2) Comparar los diferentes roles en la prueba.

1.5. Competencias Esenciales y Buenas Prácticas en la Prueba

- FL-1.5.1 (K2) Dar ejemplos de las competencias genéricas necesarias para probar.
- FL-1.5.2 (K1) Recordar las ventajas del enfoque de equipo completo.
- FL-1.5.3 (K2) Distinguir las ventajas e inconvenientes de la independencia de la prueba.

I – Fundamentos del Proceso de Prueba

1.1 ¿Qué es probar?

- Los sistemas de software son parte integrante de nuestra vida cotidiana.
- La mayoría de la gente ha tenido experiencias con software que no funcionaba como se esperaba.
- Un software que no funciona correctamente puede acarrear muchos problemas, como:
  - pérdidas económicas,
  - pérdidas de tiempo
  - o pérdidas de reputación empresarial
  - y, en casos extremos, incluso lesiones o la muerte.
- **La prueba de software evalúa la calidad del software y ayuda a reducir el riesgo de fallo del software en operación**.
- **La prueba de software es un conjunto de actividades para descubrir defectos y evaluar la calidad de los artefactos de software.**
- Cuando se prueban, estos artefactos se conocen como **objetos de prueba**.

I – Fundamentos del Proceso de Prueba

1.1 ¿Qué es probar?

Un **concepto erróneo habitual** sobre **la prueba es que sólo consiste en ejecutar pruebas**

- es decir, ejecutar el software y comprobar los resultados de las pruebas.
- Sin embargo, la prueba del software también incluye otras actividades y debe estar alineada con el ciclo de vida de desarrollo del software (véase el capítulo 2).

Otro **concepto erróneo habitual** sobre **la prueba es que ésta se concentra por completo en la verificación** del objeto de prueba.

- Aunque la prueba implica **verificación**, es decir, *comprobar si el sistema cumple los requisitos especificados*,
- también implica **validación**, lo que significa *comprobar si el sistema satisface las necesidades de los usuarios y otros implicados en su entorno operativo*.

I – Fundamentos del Proceso de Prueba

1.1 ¿Qué es probar?

- **La prueba puede ser dinámica o estática**.
  - La prueba dinámica implica la ejecución del software, mientras que la prueba estática no.
  - La prueba estática incluye revisiones (véase el capítulo 3) y análisis estático.
  - La prueba dinámica utiliza distintos tipos de técnicas de prueba y enfoques de prueba para obtener casos de prueba (véase el capítulo 4).
- **Probar no es sólo una actividad técnica**.
  - También necesita que se planifique, gestione, estime, monitorice y controle de forma adecuada (véase el capítulo 5).
- **Los probadores utilizan herramientas** (véase el capítulo 6),
  - pero es importante recordar que la prueba es en gran medida una ***actividad intelectual***, que requiere que los probadores tengan conocimientos especializados, utilicen competencias analíticas y apliquen el pensamiento crítico y el pensamiento sistémico.
- El estándar ISO/IEC/IEEE 29119-1 proporciona más información sobre los conceptos de prueba de software.

I – Fundamentos del Proceso de Prueba

1.1.1 Objetivos de Prueba

Los objetivos de prueba característicos son:

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

I – Fundamentos del Proceso de Prueba

1.1.1 Objetivos de Prueba

Los objetivos de prueba pueden **variar** en función de

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

I – Fundamentos del Proceso de Prueba

1.1.2 Probar y Depurar

Probar y depurar son actividades distintas.

- Probar:
  - puede desencadenar **fallos** causados por **defectos** en el software (**prueba dinámica**) o
  - puede, **de forma directa, encontrar defectos** en el objeto de prueba (**prueba estática**).

I – Fundamentos del Proceso de Prueba

1.1.2 Probar y Depurar

- Cuando una **prueba dinámica** (véase el capítulo 4) desencadena un fallo, la **depuración** se ocupa de:
  - encontrar las causas de este fallo (defectos),
  - analizar estas causas y eliminarlas.
- En este caso, el proceso típico de depuración implica:
  - Reproducción del fallo.
  - Diagnóstico (hallazgo de la causa raíz).
  - Corrección de la causa.
- La **prueba de confirmación** posterior comprueba si las correcciones han resuelto el problema. Preferiblemente, la prueba de confirmación la realiza la misma persona que llevó a cabo la prueba inicial.
- También se puede llevar a cabo una **prueba de regresión** posterior, para comprobar si las correcciones están causando fallos en otras partes del objeto de prueba
- (véase la sección 2.2.3 para más información sobre la prueba de confirmación y la prueba de regresión).

I – Fundamentos del Proceso de Prueba

1.1.2 Probar y Depurar

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

I – Fundamentos del Proceso de Prueba

1.1.2 Probar y Depurar

Ejemplo Objetivo de Aprendizaje (OA) FL-1.1.2

- Actividades de prueba. Aunque el presente manual describirá las actividades generales de prueba, veamos algunos ejemplos:
  - Un probador que, utilizando técnicas de diseño de casos de prueba, ejecuta los casos sobre el software para identificar defectos y fallos.
  - Un probador que analiza los documentos del software (como por ejemplo los requisitos) e identifica anomalías en sus contenidos (defectos)
- Actividades de depuración
  - El desarrollador reproduce las condiciones de uso del software que provocaron un fallo y analiza que elemento del código lo ocasiona
  - El desarrollador ejecuta el software paso a paso con una herramienta de monitorización, para analizar si el contenido de las variables está tomando los valores adecuados
- Debate: ¿Identificas más actividades de prueba y depuración? ¿Identificas alguna actividad de tu trabajo que no sabes identificar como prueba o depuración?

I – Fundamentos del Proceso de Prueba

Conceptos y definiciones

Calidad

- **Según ISTQB:**

  Grado en que un producto de trabajo satisface las necesidades enunciadas e implícitas de los implicados.

- **Según ISO 25010:**

  La calidad de un sistema es el grado en que el sistema satisface las necesidades declaradas e implícitas de sus diversas partes interesadas y, por lo tanto, proporciona valor.

- **Según ISO 9000:2015 (es):**

  Grado en el que un conjunto de características inherentes de un objeto cumple con los requisitos.

  Nota 1 a la entrada: El término "calidad" puede utilizarse acompañado de adjetivos tales como pobre, buena o excelente.

  Nota 2 a la entrada: "Inherente", en contraposición a "asignado", significa que existe en el objeto.

- **Para el examen de certificación usar exclusivamente el glosario de ISTQB**

I – Fundamentos del Proceso de Prueba

1.2 ¿Por qué es necesario probar?

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

- **¡El software como factor económico!**.

Calidad del Software. Información adicional

- La calidad del software es un factor decisivo para el éxito de determinados productos o de las propias empresas.
- Desgraciadamente todos tenemos experiencias negativas:
  - Movimientos incorrectos en la cuenta del banco, en la factura del teléfono, etc.
  - Problemas con la "centralita" del automóvil.
  - No disponibilidad de páginas Web.
  - No poder sacar dinero de la cuenta.
  - No poder realizar una gestión administrativa.
  - No poder devolver o recoger un libro, etc.

- **Nota importante**: Las páginas etiquetadas con el texto "**Información adicional**" **no** es material examinable y está fuera del alcance del programa de estudios y su correspondiente certificación. Se incluyen como información divulgativa, de conocimiento adicional y fuente para debates durante la formación.

I – Fundamentos del Proceso de Prueba
1.2 ¿Por qué es necesario probar?

Que la calidad del software sea un factor decisivo de éxito es difícil de ver, pero que la falta de calidad es un factor decisivo de fracaso, es un tema bastante claro.

Ejemplo: Fallo en la unidad de coma flotante del Pentium

OA FL-1.2.1

- En 1994 se descubrió que algunas operaciones de división devolvían siempre un valor erróneo por exceso.
- Estas comprobaciones crearon una gran polémica. Intel negó inicialmente la existencia del problema, después lo minimizó negándose a una sustitución sistemática. Si bien evaluaciones independientes mostraron la poca importancia del error llegó a haber demandas (incluyendo entre los demandantes, empresas como IBM). Por último, Intel se vio forzada a aceptar sustituir todos los microprocesadores defectuosos, lo que le representó un coste enorme.

I – Fundamentos del Proceso de Prueba
1.2 ¿Por qué es necesario probar?

Ejemplo: Phobos 1

OA FL-1.2.1

- La Phobos 1 despegó y tuvo un funcionamiento correcto hasta que dos meses después de su lanzamiento se perdió la señal. La fuente del problema fue una orden errónea (concretamente se transmitió un "+" en vez de un "-"). Incapaz de controlar su orientación, la Phobos 1 dejó de orientar sus paneles solares hacia nuestra estrella. Sin energía, no pudo restablecerse contacto con ella.
- Al margen del error, parecería lógico haber "asegurado" una orden tan crítica como demostró ser la que se envió. Estaba previsto, pero la versión definitiva del código que la contenía no se implantó a causa de las prisas en la finalización de los trabajos.

I – Fundamentos del Proceso de Prueba

1.2 ¿Por qué es necesario probar?

Ejemplo: Otros errores software famosos

OA FL-1.2.1

- Apolo 11 (fallo de alunizaje, no alunizó).
- Mariner 1 (faltaba una coma).
- Ariadne 5 (basado en una versión anterior de sw, el equipo físico no pudo responder a la mayor aceleración). Importancia de las condiciones de entorno.
- Therac-25 (Dosis masivas de radiación. Generó, al menos, 5 muertes). Importancia del control de los sistemas software .

I – Fundamentos del Proceso de Prueba

1.2.1 Contribuciones de la Prueba al Éxito

La utilización de pruebas adecuadas puede reducir la frecuencia de entregas problemáticas cuando dichas pruebas:

- Se aplican con el nivel adecuado de experiencia en prueba
- Se aplican en los niveles adecuados
- Se aplican en los puntos adecuados del ciclo de vida

- **La consecución de los objetivos de prueba definidos contribuye al éxito general del desarrollo y mantenimiento del software.**

1.2.1 Contribuciones de la Prueba al Éxito

Probar proporciona un medio rentable de detectar defectos.

- Estos defectos pueden eliminarse posteriormente (mediante la depuración, una actividad ajena a la prueba),
- por tanto, la prueba contribuye indirectamente a lograr objetos de prueba de mayor calidad.

La prueba proporciona medios para evaluar directamente la calidad de un objeto de prueba en varias etapas del ciclo de vida de desarrollo software CVDS.

- Estas mediciones se utilizan como parte de una actividad de gestión de proyectos más amplia, contribuyendo a las decisiones para pasar a la siguiente etapa del CVDS, como por ejemplo la decisión de **entrega**.

I – Fundamentos del Proceso de Prueba

1.2.1 Contribuciones de la Prueba al Éxito

Probar proporciona a los usuarios una representación indirecta en el proyecto de desarrollo.

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

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

I – Fundamentos del Proceso de Prueba

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

El aseguramiento de la calidad (AC / QA) y la prueba no son lo mismo, pero están relacionadas

I – Fundamentos del Proceso de Prueba

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

Aunque a menudo se utilizan indistintamente los términos "prueba" y "aseguramiento de la calidad" (AC / QA), **prueba y AC no son lo mismo**.

La prueba es una forma de **control de la calidad** (CC).

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

I – Fundamentos del Proceso de Prueba

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

El **aseguramiento de la calidad** es un enfoque **preventivo** orientado a los procesos que se concentra en la implementación y mejora de los mismos.

- Funciona sobre la base de que, si un buen proceso se sigue correctamente, entonces generará un buen producto.
- El **aseguramiento de la calidad** se aplica tanto al proceso de desarrollo como al de prueba, y es responsabilidad de todos los que participan en un proyecto.

Los resultados de la prueba son utilizados por el Aseguramiento de la Calidad (AC) y el Control de la Calidad (CC).

- En el control de calidad (CC) se utilizan para corregir defectos,
- mientras que el aseguramiento de la calidad (AC) proporciona retroalimentación sobre el rendimiento de los procesos de desarrollo y prueba.

I – Fundamentos del Proceso de Prueba

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

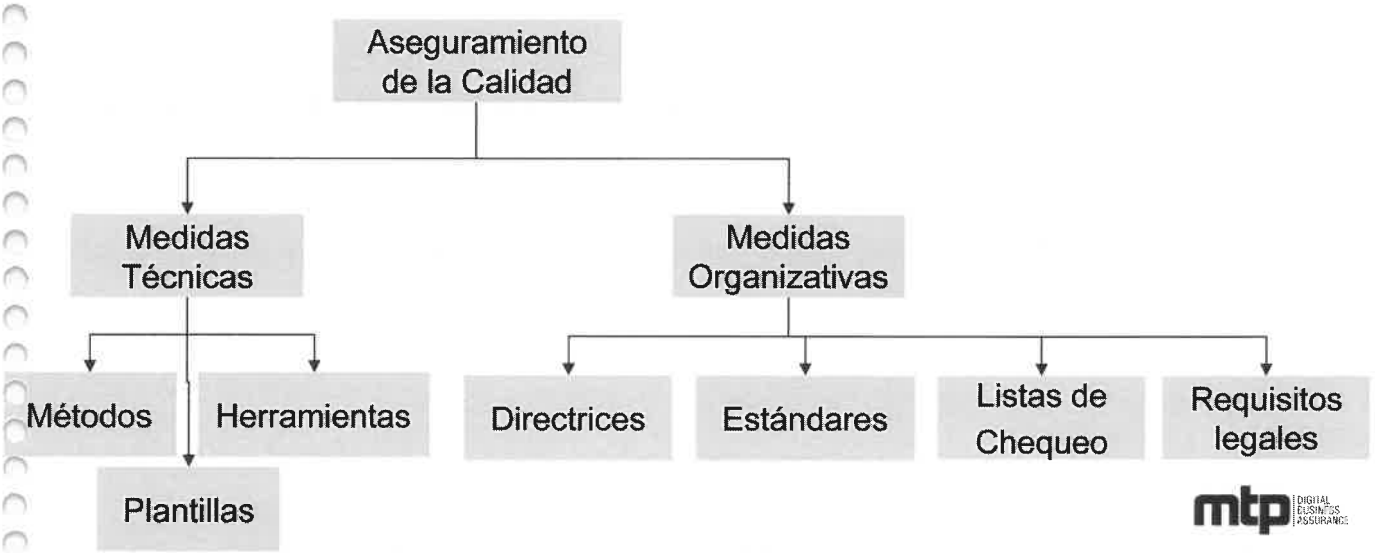
Conceptos. Información adicional:

- **Gestión de la Calidad**: incluye todas las actividades que dirigen y controlan una organización con respecto a la calidad. Incluye al aseguramiento de la calidad y el control de la calidad
- **Aseguramiento de la Calidad**: se centra, por lo general, en el cumplimiento de los procesos adecuados, a fin de proporcionar la confianza de que se alcanzarán los niveles de calidad adecuados. Aspectos importantes:
  - Con procesos realizados de forma correcta, los productos de trabajo son generalmente de mayor calidad
  - Esto contribuye a la **prevención** de defectos
  - El **análisis de causa raíz** detecta y elimina la causa de los defectos
  - Promueve la prueba adecuada (saber cómo probar)
- **Control de la Calidad**: implica varias actividades en las que se incluyen las actividades de prueba que sirven para alcanzar los niveles de calidad apropiados. Está **integrado** con el proceso general de desarrollo y mantenimiento del software

I – Fundamentos del Proceso de Prueba

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

Aseguramiento de la Calidad. Información adicional

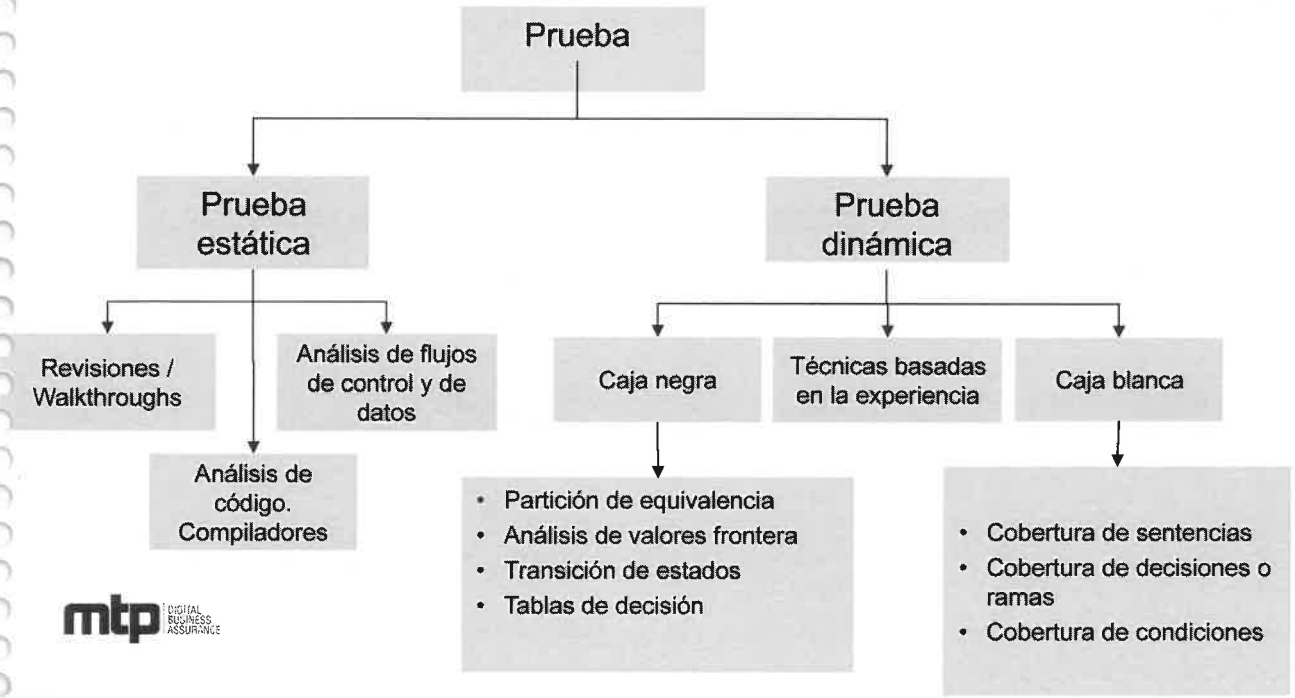


- Motivación principal:
  Los defectos que no se inyectan en el producto no necesitan ser corregidos.

I – Fundamentos del Proceso de Prueba

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

Control de Calidad. Información adicional



I – Fundamentos del Proceso de Prueba

1.2.2 Prueba y Aseguramiento de la Calidad (AC)

Las Pruebas como medio de mejora de la calidad. Información adicional

- Un medio para conseguir la mejora de la calidad tanto de los sistemas de software como del propio proceso de desarrollo es la comprobación y prueba sistemática del software desarrollado.
- Los errores que se detecten antes del uso del software pueden ser corregidos antes de que generen fallos.
- Puede exigirse por contrato un nivel mínimo de prueba.
- Las pruebas pueden requerirse también para satisfacer requisitos contractuales o legales, o estándares específicos de la industria.

I – Fundamentos del Proceso de Prueba

1.2.3 Errores, Defectos, Fallos y Causas Raíz

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

1.2.3 Errores, Defectos, Fallos y Causas Raíz

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

I – Fundamentos del Proceso de Prueba

1.2.3 Errores, Defectos, Fallos y Causas Raíz

- Los errores y los defectos no son la única causa de los **fallos**.
  - Los fallos también pueden deberse a condiciones ambientales, como cuando la radiación o el campo electromagnético provocan defectos en el firmware.
- Una **causa raíz** es una razón fundamental por la que se produce un problema (por ejemplo, una situación que conduce a un error).
  - Las causas raíz se identifican mediante el **análisis de la causa raíz**, que suele realizarse cuando se produce un fallo o se identifica un defecto.
  - Se cree que pueden evitarse otros fallos o defectos similares o reducirse su frecuencia tratando la causa raíz, por ejemplo, eliminándola.

I – Fundamentos del Proceso de Prueba

1.2.3 Errores, Defectos, Fallos y Causas Raíz

Fallos, defectos, errores…

- **Equivocación** (mistake):

  Acción humana que da lugar a un resultado incorrecto.

- **Defecto, falta** (bug, defect, fault):

  Anomalía en un componente o sistema que puede dar lugar a que no se lleve a cabo correctamente una función determinada. El término "bug" se aplica históricamente a los defectos en informática

- **Fallo** (failure). Efecto del defecto

  Manifestación de un defecto. Desviación entre el comportamiento real y el comportamiento esperado, no cumplimiento de un requisito establecido

- **Error**:

  Ambiguo. Sinónimo de equivocación en esta certificación

El software es elaborado por seres humanos que pueden cometer errores que dan lugar a defectos. Estos defectos pueden generar un fallo.

I – Fundamentos del Proceso de Prueba

1.2.3 Errores, Defectos, Fallos y Causas Raíz

Causas raíz de los defectos: acciones o condiciones más tempranas que contribuyeron a crear los defectos. Proceso:

- Analizando los defectos se puede encontrar su causa raíz
- Poniendo foco en las causas raíz más significativas, se puede realizar mejoras en el proceso de trabajo previniendo la introducción de futuros defectos (mejora de procesos)

Ejemplo OA FL-1.2.3: analizando los defectos del pago de intereses incorrectos que provocaron reclamaciones de clientes, se puede encontrar defectos similares que evidencien malentendidos con los propietarios del producto.

- **Efectos**: las reclamaciones de los clientes
- **Fallos**: pagos de intereses de forma incorrecta
- **Causa raíz**: falta de conocimiento del propietario del producto
- **Defecto**: historia de usuario no adecuada
- **Error**: El malentendido del propietario de producto, por falta de experiencia

I – Fundamentos del Proceso de Prueba

1.3 Siete Principios de la Prueba

Principios de la prueba: Directrices generales aplicables a todas las pruebas. Estas directrices se han obtenido de la experiencia desde los años 70 del pasado siglo

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

- La prueba puede mostrar que hay defectos en el objeto de prueba, pero **no puede demostrar que no los haya**.
- La prueba reduce la probabilidad de que queden defectos por descubrir en el objeto de prueba, pero, aunque no se encuentren defectos, **la prueba no puede demostrar la corrección del objeto de prueba**.
- **La ausencia de fallos no prueba que el software sea correcto**
  - El propio procedimiento de prueba puede contener defectos
  - Las condiciones de pruebas pueden no estar bien preparadas para encontrar defectos

I – Fundamentos del Proceso de Prueba

1.3 Siete Principios de la Prueba

Principio 2. La prueba exhaustiva es imposible.

- Probarlo todo no es factible salvo en casos triviales.
- En lugar de intentar probar exhaustivamente, para concentrar los esfuerzos de prueba deberían utilizarse:
  - técnicas de prueba (véase el capítulo 4),
  - priorización de casos de prueba (véase el apartado 5.1.5)
  - y pruebas basadas en el riesgo (véase el apartado 5.2).

Definiciones

- **Pruebas exhaustivas**. Es una estrategia de pruebas en la que se abarcan todas las posibles combinaciones de valores de entrada y precondiciones.
- **Explosión de casos de prueba**. El crecimiento desproporcionado del número de casos de prueba con el tamaño creciente de la base de prueba, cuando se utiliza una determinada técnica de diseño de prueba. La explosión de casos de prueba también puede producirse al aplicar sistemáticamente una técnica de diseño de pruebas por primera vez..

Principio 3. La prueba temprana ahorra tiempo y dinero.

- Los defectos que se eliminan al principio del proceso no causarán defectos posteriores en los productos de trabajo derivados.
- El coste de la calidad se reducirá, ya que se producirán menos fallos más adelante en el CVDS.
- Para encontrar defectos en una fase temprana, tanto **las pruebas** estáticas (véase el capítulo 3) como las pruebas dinámicas (véase el capítulo 4) **deben iniciarse lo antes posible**.
- Sinónimo de prueba temprana: **desplazamiento hacia la izquierda (shift left)**

Principio 3. Información adicional.

- Las actividades de prueba deberían iniciarse tan pronto como sea posible en el ciclo de vida de desarrollo y deberían tener objetivos concretos.
- Cuanto antes se descubra un defecto, menos costosa será su corrección.
  - La máxima efectividad se alcanza cuando los defectos se corrigen antes de ser implementados (p.e. insertados en código).
  - Pueden probarse (revisarse) los documentos de requisitos conceptuales y las especificaciones.
  - Los defectos que se descubren en la fase de concepción del proyecto se corrigen con mínimos esfuerzos y costes.
- La preparación de una prueba consume tiempo. La prueba no es sólo la ejecución.
  - Pueden realizarse actividades de prueba (Diseño) antes de que se complete el desarrollo del software.
  - Las actividades de prueba (considerando como tales las revisiones) deberían llevarse a cabo en paralelo a la especificación y el diseño del software.

Principio 4. Los defectos se agrupan.

- Un pequeño número de componentes del sistema suelen contener la mayoría de los defectos descubiertos o son responsables de la mayoría de los fallos operativos.
- Este fenómeno es una ilustración del **principio de Pareto**.
- Las agrupaciones de defectos previstas, y las agrupaciones de defectos reales observadas durante la prueba o en operación, son una entrada importante para la prueba basada en el riesgo (véase la sección 5.2).

Principio 4. Información adicional.

- Encuentre un defecto y encontrará más en los alrededores
  - Los defectos aparecen frecuentemente en grupos como las setas.
  - Es una buena idea repasar el módulo en el que se ha encontrado un defecto.
- Los probadores deben ser flexibles
  - Una vez detectado un defecto, es una buena idea reconsiderar la dirección de las siguientes pruebas.
  - La localización de un defecto debe ser revisada con un mayor nivel de detalle, ya sea incorporando nuevas pruebas o modificando las existentes.

Principio 5. Las pruebas se desgastan.

- Si las pruebas se repiten muchas veces, se vuelven cada vez más ineficaces para detectar nuevos defectos.
- Para superar este efecto, puede que sea necesario modificar las pruebas y los datos de prueba existentes, y que haya que escribir nuevas pruebas.
- Sin embargo, en algunos casos, la repetición de las mismas pruebas puede tener un resultado beneficioso, por ejemplo, en las pruebas de regresión automatizadas (véase la sección 2.2.3).

Principio 5. Información adicional.

- Cualquier método que se use para prevenir o encontrar defectos deja un residuo o defectos más sutiles, contra los que ese método no es efectivo.
- Repetir las pruebas bajo las mismas condiciones no es efectivo.
  - Si se repiten una y otra vez las mismas pruebas no se encontrarán nuevos errores.
- Las pruebas deben ser revisadas regularmente.
  - Es necesario repetir una prueba después de que se hagan cambios en el código.
  - Las pruebas automatizadas pueden ser una ventaja si se usa regularmente un grupo de casos de prueba.

I – Fundamentos del Proceso de Prueba

1.3 Siete Principios de la Prueba

Principio 6. La prueba depende del contexto.

- No existe un único enfoque de prueba aplicable universalmente.
- La prueba se realiza de forma diferente en distintos contextos.

- **Información adicional**:
  - Una prueba en un proyecto ágil se realiza de forma distinta a la que se realiza en un ciclo de vida secuencial
  - Las pruebas pueden tener distintos resultados en distintos contextos.
    - Diferentes objetos de prueba se prueban de forma distinta.
    - El controlador del motor de un coche requiere distintas pruebas que las de una aplicación de comercio electrónico.
  - Pruebas en el entorno de prueba vs pruebas en el entorno de producción.
    - Las pruebas deben llevarse a cabo en un entorno separado al de producción. El entorno de prueba debe ser muy similar al de producción.
    - Siempre habrá desviaciones entre ambos entornos.

I – Fundamentos del Proceso de Prueba

1.3 Siete Principios de la Prueba

Principio 7. Falacia de la ausencia de defectos.

- *Es una falacia (es decir, una idea equivocada) esperar que la verificación del software asegure el éxito de un sistema.*
- **Probar a fondo** todos los requisitos especificados y arreglar todos los defectos encontrados …
- … podría seguir produciendo un sistema
  - que no satisface las necesidades y expectativas de los usuarios,
  - que no ayuda a conseguir los objetivos de negocio del cliente y
  - que es inferior en comparación con otros sistemas de la competencia.
- Además de la **verificación**, también debe llevarse a cabo la **validación**.

I – Fundamentos del Proceso de Prueba

1.3 Siete Principios de la Prueba

Principio 7. Información adicional.

- Unas pruebas adecuadas localizan los fallos más serios.
  - En la mayoría de los casos, las pruebas no encontrarán todos los defectos del sistema (ver principio 2) pero sí los más serios.
- Con esto solamente, no se prueba la calidad del software.
  - La funcionalidad del software puede no cumplir las necesidades y expectativas de los usuarios.
  - No se puede probar la calidad de un producto **solo con las pruebas**, debe construirse desde el principio con la calidad requerida.

I – Fundamentos del Proceso de Prueba

1.3 Siete Principios de la Prueba

Ejemplos y dinámicas sobre los principios de las pruebas:

OA FL-1.3.1

- Principio 1: Identifiquemos situaciones de nuestros proyectos en las que el software o el producto falló estando convencidos de haberlo "probado bien"
- Principio 2: Cuanta más experiencia profesional se tiene, más fácil es comprender este principio. Sin embargo, existe una comprobación rápida: después de finalizar la presente formación, intenta planificar todas las propuestas de actividades de prueba que contiene este manual. ¿lo ves viable? Y eso que es una formación básica …
- Principio 3: Existen muchas situaciones cotidianas (incluso a ajenas a la creación de software) donde se produce la situación de "más vale prevenir que curar". Reflexionemos sobre situaciones de nuestros proyectos en los que hayamos vivido esta situación.

I – Fundamentos del Proceso de Prueba

1.3 Siete Principios de la Prueba

Ejemplos y dinámicas sobre los principios de las pruebas:

OA FL-1.3.1

- Principio 4: Aunque posiblemente todos conozcamos el principio de Pareto, identifiquemos situaciones de nuestros proyectos en los que unos pocos componentes han causado el mayor número de problemas
- Principio 5: Identifiquemos situaciones de nuestros proyectos que nuestras pruebas "habituales" no detectaron fallos posteriores ¿por qué? ¿Puede estar relacionado con el principio 2?. Una situación habitual relacionada con este principio son las carencias en pruebas de mantenibilidad.
- Principio 6: ¿Cuántas veces un software o producto paso las pruebas en preproducción y falló en producción? ¿por qué? ¿qué cambió?. Lo que es adecuado en una situación no tiene que serlo para todas
- Principio 7: Por desgracia este principio ocurre demasiadas veces. Pensemos en situaciones de nuestros proyectos que estaban "bien desarrolladas y probadas" y el cliente o usuario indicó no estar completamente satisfecho. ¿por qué?

I – Fundamentos del Proceso de Prueba

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

La prueba depende del contexto, pero, …

… a un alto nivel, existen conjuntos comunes de actividades de prueba sin los cuales es menos probable que la prueba alcance los objetivos de prueba.

Estos conjuntos de actividades de prueba forman un proceso de prueba.

El proceso de prueba puede adaptarse a una situación determinada en función de diversos factores.

Qué actividades de prueba se incluyen en este proceso de prueba, cómo se implementan y cuándo tienen lugar (**estrategia de prueba**) se decide normalmente como parte de la **planificación de prueba** para la situación específica (véase la sección 5.1).

I – Fundamentos del Proceso de Prueba

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

Las siguientes secciones describen los aspectos generales de este proceso de prueba en términos de

- actividades y tareas de prueba,
- el impacto del contexto,
- el producto de prueba,
- la trazabilidad entre la base de prueba y el producto de prueba,
- y los roles de prueba.

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

Concepto (según glosario oficial del ISTQB):

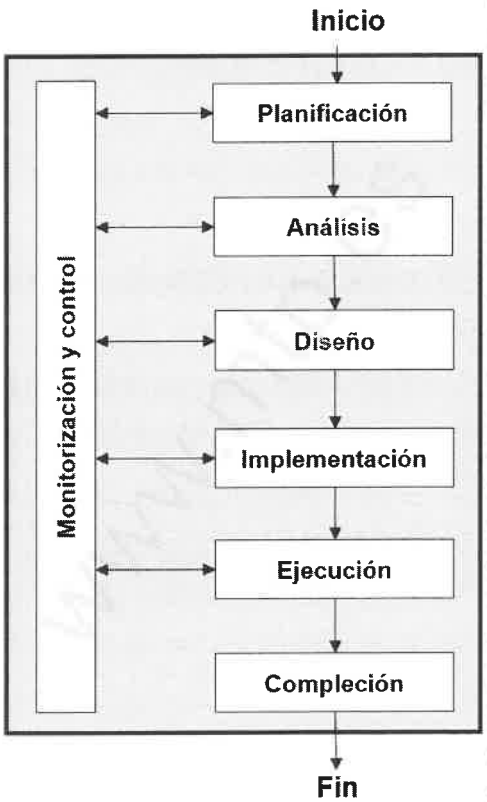
Cobertura: Grado en que los **elementos de cobertura** especificados son practicados por un **juego de pruebas**, expresado como porcentaje.

I – Fundamentos del Proceso de Prueba

1.4.1 Actividades y Tareas de Prueba

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

- **¡¡Siempre hay que adaptar estas actividades al contexto del sistema o proyecto!!**



I – Fundamentos del Proceso de Prueba

1.4.1 Actividades y Tareas de Prueba

Planificación de la Prueba.

- La planificación de la prueba consiste en definir los objetivos de la prueba y, a continuación, seleccionar el **enfoque** que mejor permita alcanzar los objetivos dentro de las limitaciones impuestas por el contexto general.
- La planificación de la prueba se explica con más detalle en la sección 5.1.
- **Ejemplos de tareas de planificación** OA FL-1.4.1:
  - Identificar qué probar y qué no probar
  - Asignar profesionales a las tareas
  - Identificar las necesidades de entornos y las herramientas
  - Comprobar que tenemos recursos y tiempo para las actividades
  - Seleccionar las técnicas y tipo de prueba
  - ¿Puedes poner algún ejemplo más de tu proyecto?

I – Fundamentos del Proceso de Prueba

1.4.1 Actividades y Tareas de Prueba

Monitorización y Control de la Prueba.

- La monitorización de la prueba implica la comprobación continua de todas las actividades de la prueba y la comparación del avance real con el plan.
- El control de prueba implica la adopción de las medidas necesarias para cumplir los objetivos de la prueba.
- La monitorización de prueba y el control de prueba se explican con más detalle en la sección 5.3.
- **Ejemplo** OA FL-1.4.1:
  - Las actividades relacionadas con la elaboración de los informes de prueba

1.4.1 Actividades y Tareas de Prueba

Análisis de la Prueba.

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

1.4.1 Actividades y Tareas de Prueba

Diseño de la Prueba.

El diseño de prueba incluye la elaboración de las condiciones de prueba en **casos de prueba** y otros productos de prueba (por ejemplo, contratos de prueba).

Esta actividad suele implicar la identificación de elementos de **cobertura**, que sirven de referencia para especificar las entradas de los casos de prueba. Las técnicas de prueba (véase el capítulo 4) pueden servir de apoyo a esta actividad.

El diseño de pruebas también incluye

- la definición de los requisitos de **datos de prueba**,
- el diseño del **entorno de prueba**
- y la identificación de cualquier otra **infraestructura y herramientas** necesarias.

El diseño de prueba responde a la pregunta **"¿cómo llevar a cabo la prueba?"**.

Ejemplos de Análisis y Diseño de prueba OA FL-1.4.1.

- La lectura de la documentación (p.e. requisitos)
- La lectura y análisis a nivel de equipo de una historia de usuario
- Aplicación de técnicas de diseño de casos de pruebas (p.e. las descritas en el capítulo 4)
- Realización de actividades de revisión de documentación (p.e. las descritas en el capítulo 3)
- ¿Se pueden identificar ejemplos de actividades de análisis y diseño de casos de pruebas que se apliquen en tu proyecto?

Implementación de la Prueba.

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

Ejecución de la Prueba.

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

Compleción de la Prueba.

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

I – Fundamentos del Proceso de Prueba

1.4.1 Actividades y Tareas de Prueba

Ejemplos de Implementación, Ejecución y Compleción de la Prueba.
OA FL-1.4.1

- Instalación del entorno y herramientas de prueba
- Creación de los scripts (p.e. código) de automatización de la prueba
- Selección, anonimización, archivo de datos de prueba
- Registro de los resultados de la ejecución de los casos de prueba
- Entrega de los informes de avance de la prueba
- Entrega de los informes resumen de la prueba
- Evaluación grupal de cómo mejorar las pruebas
- ¿Qué diferencias existen entre las actividades relativas a los entornos y herramientas de prueba durante la planificación y durante la implementación de la prueba?
- ¿Identificas más actividades de tus proyectos que podrían ser ejemplos adicionales?

I – Fundamentos del Proceso de Prueba

1.4.2 El Proceso de Prueba en Contexto

- La prueba no se realiza de forma aislada.
- Las actividades de prueba son parte **integrante** de los procesos de desarrollo que se llevan a cabo en una organización.
- La prueba también está sufragada por los implicados y su objetivo final es ayudar a satisfacer las **necesidades de negocio** de los implicados.
- Por lo tanto, la forma en que se lleve a cabo la prueba dependerá de una serie de factores contextuales
- Estos factores influirán en muchos asuntos relacionados con la prueba, entre ellos:
  - la estrategia de prueba,
  - las técnicas de prueba utilizadas,
  - el grado de automatización de la prueba,
  - el nivel de cobertura requerido,
  - el nivel de detalle de la documentación de prueba,
  - el suministro de información, etc.

1.4.2 El Proceso de Prueba en Contexto

Factores contextuales que influyen en la forma de realizar la prueba:

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

I – Fundamentos del Proceso de Prueba

1.4.2 El Proceso de Prueba en Contexto

Ejemplos OA FL-1.4.2:

- El software tiene algunas faltas de ortografía en el texto ¿Sería una situación muy grave? ¿Y si fuese en la https://www.rae.es/?
- El software aplica incorrectamente el redondeo en el último dígito de los céntimos del importe final. ¿La gravedad sería la misma para una pequeña tienda de barrio que para un sistema bancario?

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

I – Fundamentos del Proceso de Prueba
1.4.3 Productos de Prueba

Productos del Trabajo de Planificación de la Prueba

- Entre los productos del trabajo de planificación de la prueba se incluyen:
  - el plan de prueba,
  - el calendario de prueba,
  - el registro de riesgos
  - y los criterios de entrada y salida (véase la sección 5.1).
- El **registro de riesgos** es una lista de riesgos junto con la probabilidad del riesgo, el impacto del riesgo e información sobre la mitigación del riesgo (véase la sección 5.2).
- *El calendario de prueba, el registro de riesgos y los criterios de entrada y salida suelen formar parte del **plan de prueba**.*

I – Fundamentos del Proceso de Prueba

1.4.3 Productos de Prueba

Ejemplo OA FL-1.4.3:

- Veamos los productos de trabajo de prueba más habituales en una herramienta de prueba muy extendida https://testlink.org/
- ¿Identificas otros productos de trabajo en tus proyectos? ¿Podrías clasificarlos?

I – Fundamentos del Proceso de Prueba

1.4.4 Trazabilidad entre Bases de Prueba y Productos de Prueba

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

1.4.4 Trazabilidad entre Bases de Prueba y Productos de Prueba

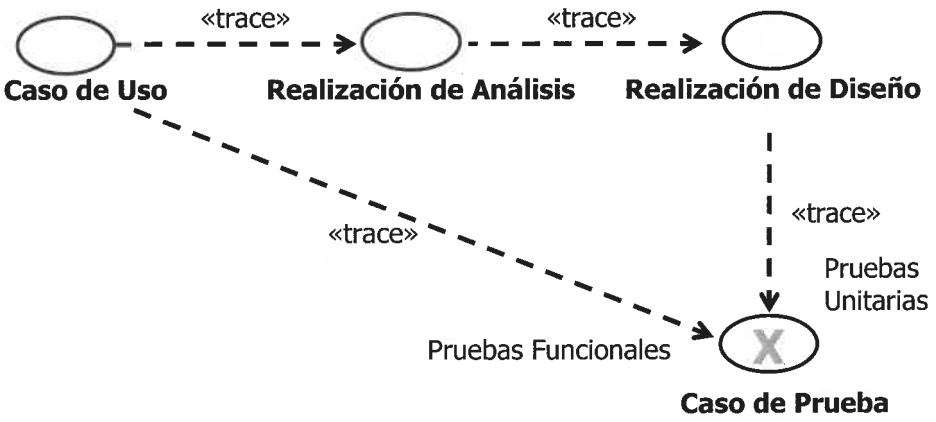
- Una trazabilidad precisa soporta la **evaluación de la cobertura**, por lo que es muy útil que se definan criterios de cobertura medibles en la base de prueba.
- Los criterios de cobertura pueden funcionar como **indicadores clave de rendimiento** (KPIs) para impulsar las actividades que muestran hasta qué punto se han alcanzado los objetivos de la prueba (véase la sección 1.1.1). Por ejemplo:
  - La trazabilidad de los casos de prueba a los requisitos puede verificar que los requisitos están cubiertos por los casos de prueba.
  - La trazabilidad de los resultados de prueba a los riesgos puede utilizarse para evaluar el nivel de riesgo residual en un objeto de prueba.

I – Fundamentos del Proceso de Prueba
1.4.4 Trazabilidad entre Bases de Prueba y Productos de Prueba

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

I – Fundamentos del Proceso de Prueba

1.4.4 Trazabilidad entre Bases de Prueba y Productos de Prueba



[The Unified Software Development Process. I. Jacobson, G. Booch and J. Rumbaugh. Addison-Wesley, 1999]

Ejemplo OA FL-1.4.4:

- Veamos cómo se aplica la trazabilidad y su utilidad en https://testlink.org/

I – Fundamentos del Proceso de Prueba

1.4.5 Roles en la Prueba

En este programa de estudio se tratan dos roles principales en la prueba:

- un **rol de gestión de prueba** y
- un **rol de prueba**.

Las actividades y tareas asignadas a estos dos roles dependen de factores como:

- el contexto del proyecto y del producto,
- las competencias de las personas que desempeñan los roles y
- la organización.

I – Fundamentos del Proceso de Prueba

1.4.5 Roles en la Prueba

El **rol de gestión de prueba asume la responsabilidad general**

- del proceso de prueba,
- del equipo de prueba y de la dirección
- de las actividades de prueba.

El rol de gestión de prueba se concentra principalmente en las actividades de **planificación** de prueba, **monitorización y control** de prueba y **compleción** de la prueba.

La forma en que se lleva a cabo el rol de gestión de pruebas varía en función del contexto.

- **Por ejemplo** OA FL-1.4.5,
  - en el desarrollo ágil de software, algunas de las tareas de gestión de pruebas pueden correr a cargo del equipo ágil.
  - Las tareas que abarcan a varios equipos o a toda la organización pueden ser realizadas por jefes de prueba ajenos al equipo de desarrollo.

I – Fundamentos del Proceso de Prueba

1.4.5 Roles en la Prueba

El **rol de prueba asume la responsabilidad general del aspecto asociado a la ingeniería (técnica) de la prueba**.

El rol de prueba se concentra principalmente en las actividades de **análisis** de prueba, **diseño** de prueba, **implementación** de prueba y **ejecución** de prueba.

Diferentes personas pueden asumir estos roles en diferentes momentos.

- **Por ejemplo** OA FL-1.4.5,
  - el rol de gestión de la prueba puede ser desempeñado por un jefe de equipo, por un jefe de prueba, por un jefe de desarrollo, etc.
  - También es posible que una misma persona asuma los roles de prueba y gestión de prueba al mismo tiempo.

I – Fundamentos del Proceso de Prueba

1.4.5 Roles en la Prueba

Tareas de un Jefe de Prueba. Información adicional y ejemplos

- Se necesitan experiencias especiales en las áreas de:
  - Pruebas de SW y gestión de la calidad.
  - Planificación y dirección de pruebas.
  - Experiencia en dirección de proyectos.
  - Capacidad para dirigir.

I – Fundamentos del Proceso de Prueba

1.4.5 Roles en la Prueba

Perfiles. Información adicional y ejemplos

- Para **proyectos grandes, complejos o críticos** desde el punto de vista de la seguridad de las personas, suele ser mejor tener **múltiples niveles de prueba**, realizándose algunos o todos ellos por probadores independientes.
- El personal de **desarrollo** puede participar en las pruebas, especialmente en los niveles más bajos, pero su falta de objetividad suele limitar su efectividad.
- Los probadores independientes pueden tener autoridad para requerir y definir procesos y reglas de pruebas pero los probadores sólo deberían asumir esos roles relacionados con el proceso sólo cuando hay un claro mandato al respecto.
- Dependiendo del nivel de prueba y los riesgos relacionados con el producto y el proyecto, diferentes personas pueden asumir el papel de probador manteniendo un cierto grado de independencia.
- Los probadores típicos a nivel de componente y de integración de componentes serían los desarrolladores, en las pruebas de aceptación, expertos en negocio y usuarios y en la aceptación operativa los operadores.
- Las pruebas de sistema e integración de sistemas suelen ser llevadas a cabo por equipos independientes de prueba.

Perfiles. Información adicional y ejemplos

- Respecto a los probadores, en función del grado de especialización se puede hablar de:
  - Analista de pruebas.
  - Diseñador de pruebas.
  - Automatizador de pruebas.
  - Administrador del sistema de pruebas.
  - Ejecutor de pruebas.
- Se pueden diferenciar otros roles.

I – Fundamentos del Proceso de Prueba
1.4.5 Roles en la Prueba

Perfiles. Información adicional y ejemplos

Analista y diseñador de pruebas.

- Estudia la documentación de entrada con el fin de analizar, revisar y evaluar los requisitos de usuario, las especificaciones y los modelos, para valorar qué tanto pueden ser probados.
- Propone las pruebas necesarias y establece el orden de su ejecución.
- Conocimientos:
  - Conocimiento (Know-how) de desarrollo y pruebas.
  - Conocimientos de ingeniería de SW.
  - Conocimientos sobre métodos de análisis y de especificación.

Automatizador de pruebas.

- Comprueba las posibilidades de automatización y las lleva a cabo.
- Conocimientos:
  - Experiencia como probador.
  - Conocimientos (Know-how) en diseño de pruebas y automatización.
  - Experiencia en programación.
  - Muy buenos conocimientos de las herramientas implantadas.

Perfiles. Información adicional y ejemplos

Administrador del sistema de pruebas.

- Implanta el entorno de las pruebas y lo gestiona.
- Conocimientos:
  - Administración de sistemas (o acceso a un administrador de sistemas).
  - Herramientas de desarrollo y pruebas.
  - Sistemas de bases de datos, Redes en caso de ser necesario.
  - Instalación y gestión del entorno del sistema.

Ejecutor de las pruebas de software.

- Ejecuta las pruebas en función de los supuestos / especificaciones.
- Conocimientos:
  - Conocimientos generales de IT y básicos de pruebas.
  - Manejo / parametrización de la herramienta implantada.
  - Experiencia en la ejecución de pruebas.
  - Conocimientos acerca del objeto de prueba.

I – Fundamentos del Proceso de Prueba
1.4.5 Roles en la Prueba

Perfiles. Información adicional y ejemplos

Experto técnico.

- Completa el equipo de pruebas en caso de necesidad.
- Pueden ser:
  - Administradores o diseñadores de bases de datos.
  - Expertos en interfases de usuario.
  - Especialistas en redes.

Otros perfiles.

- Según el planteamiento del problema, el entorno de pruebas, etc. pueden incorporarse otros especialistas al equipo de pruebas.
  - Aquí se necesitan conocimientos especiales en materias que no tengan relación con las pruebas, p.ej., expertos en usabilidad o psicólogos.

Perfiles. Información adicional y ejemplos

Perfiles Soft

- A las cualificaciones técnicas referidas se añaden factores como:
  - Capacidad de trabajo en equipo, habilidad política o diplomática.
  - Disponibilidad para consultar acerca de hechos evidentes.
  - Capacidad para imponerse, apariencia de seguridad.
  - Exactitud y creatividad.
  - Manejar con seguridad situaciones complejas.

Comentario:
Sin estas características añadidas un probador sólo tendrá un éxito condicional aunque sea muy competente en la materia.

Reflexión. Información adicional.

El desarrollo de software, incluyendo la prueba de software, involucra a seres humanos. Por lo tanto, las características de los seres humanos tienen efectos importantes en el desarrollo y la prueba de software.

Las personas y los proyectos se conducen en base a objetivos. Las personas tienden a alinear sus planes con los objetivos establecidos por los gestores u otros responsables, por ejemplo: encontrar defectos o confirmar que funciona el software.

Por eso es importante establecer de una forma clara los **objetivos de las pruebas**.

I – Fundamentos del Proceso de Prueba

1.5 Competencias Esenciales y Buenas Prácticas de la Prueba

La competencia es la capacidad de hacer algo bien que proviene

- del conocimiento,
- la práctica y
- la aptitud de cada uno.

Los buenos probadores deben poseer algunas **competencias esenciales** para hacer bien su trabajo.

Los buenos probadores deben ser

- jugadores de **equipo** eficaces y
- deben ser capaces de realizar pruebas en diferentes **niveles de independencia** de la prueba.

I – Fundamentos del Proceso de Prueba

1.5.1 Competencias Genéricas Necesarias para Probar

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

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

Adicionalmente:

- A menudo, los probadores son los portadores de las **malas noticias**.
- Es un rasgo humano común **culpar** al portador de malas noticias.
- Esto hace que las **competencias de comunicación** sean cruciales para los probadores.
- Comunicar los resultados de prueba puede percibirse como una **crítica** al producto y a su autor.
- El **sesgo de confirmación** puede dificultar la aceptación de información que discrepa de las creencias actuales.
- Algunas personas pueden percibir las pruebas como una **actividad destructiva (¡NO!)**, a pesar de que contribuyen en gran medida al éxito del proyecto y a la calidad del producto.
- Para intentar mejorar esta opinión, la información sobre defectos y fallos debe comunicarse de **forma constructiva**.

Ejemplo OA FL-1.5.1:

- El conocimiento en materia de pruebas (cómo este curso) otorga al probador la capacidad de probar lo adecuado (ni mucho ni poco). ¿En qué ámbitos cotidianos podemos observar la importancia de "ni quedarse corto ni pasarse"?
- Una comunicación inadecuada afecta negativamente al proyecto. Muchos de los problemas que sufrimos tienen un origen habitual en una insuficiente comunicación. Si aparecen frases como "Es que pensé que …", "Yo creí que …", "Entendí que …" es posible que la comunicación deba mejorar
- En ocasiones, los equipos de desarrollo nos dicen cosas como "no se me había ocurrido usar el software de esa manera". En ocasiones, al centrarse en la construcción de un producto, se nos olvida las diferentes formas de uso. Por ejemplo, construir un camino peatonal asfaltando y comprobar que las personas transitan por otro lado.
- ¿Tienes más situaciones como estas en tus proyectos?

Personalidad de un buen probador. Información adicional

- Curioso, atento a los detalles.
  - Para comprender los escenarios prácticos en los que se mueve el cliente.
  - Para ser capaz de analizar la estructura de la prueba.
  - Para descubrir detalles que pueden identificar fallos.
- Escepticismo y ojo crítico.
  - Los objetos de prueba contienen defectos. Se trata de encontrarlos.
  - No se debe ver limitado por el hecho de que un error grave pueda afectar a (por ejemplo) las fechas del proyecto.
- Buenas capacidades de comunicación.
  - Para dar malas noticias a los desarrolladores.
  - Para no perder la perspectiva ante situaciones frustrantes.
  - Para establecer rápidamente una relación de trabajo con los desarrolladores.
- La comunicación positiva puede ayudar a evitar o facilitar situaciones complicadas.
- Experiencia.
  - Los factores personales influyen en la ocurrencia de errores.
  - La experiencia ayuda a identificar donde se pueden acumular los errores.

I – Fundamentos del Proceso de Prueba
1.5.1 Competencias Genéricas Necesarias para Probar

Problemas de comunicación. Información adicional

- Especialmente en fases del proyecto estresantes la notificación de los errores puede generar problemas, sobre todo si los probadores son vistos como portadores de malas noticias.
- La manera de notificar el error y la descripción del mismo son decisivos.
- No criticar a las personas sino mostrar el error de forma objetiva.
- Facilitar la solución del error al desarrollador mediante la notificación.
- El objetivo común debe estar siempre en primer plano.
- Una comunicación insuficiente o la ausencia de ésta entre probadores y desarrolladores puede impedir el buen trabajo en equipo.
- En ocasiones, no hay entendimiento entre el equipo de pruebas y el de desarrollo.
  - Los desarrolladores deberían tener nociones básicas de pruebas.
  - Los probadores deberían conocer las características esenciales del desarrollo de software

Problemas de comunicación. Información adicional

- Hay, sin embargo, varias formas de mejorar la comunicación y las relaciones entre los probadores y el resto:
  - Empezar a colaborar, en vez de entrar en refriegas - recordar a todos que el objetivo común es obtener sistemas de mejor calidad.
  - Comunicar lo que se ha encontrado en el producto de forma neutra, enfocada a los hechos, sin criticar a la persona que lo ha creado, por ejemplo, redactando informes de incidencias y revisiones de forma objetiva y ateniéndose a los hechos.
  - Intentar comprender como se siente la otra persona y por qué reacciona como lo hace.
  - Confirmar que la otra persona ha entendido lo que has dicho y viceversa.

Información adicional. La mentalidad a utilizar durante las pruebas y revisiones es distinta a la que se tiene durante el desarrollo.

El desarrollador.

- Transforma los requisitos.
- Desarrolla estructuras.
- Programa el software.
- Consigue un producto.

⇒ ¡El desarrollador es constructivo!.

El probador.

- Planifica sus pruebas.
- Especifica casos de prueba.
- Quiere encontrar defectos.
- Los errores del programador son su éxito.

⇒ ¡El probador es destructivo!.

¡Falso!

¡También las pruebas son constructivas, ya que su objetivo es un producto de calidad adecuada!

1.5.2 Enfoque de Equipo Completo

Una de las competencias importantes para un probador es la capacidad de **trabajar eficazmente en un contexto de equipo** y de contribuir positivamente a los objetivos del mismo.

El **enfoque de equipo completo** se basa en esta competencia. Es una práctica procedente de la Programación Extrema (véase la sección 2.1)

En el enfoque de equipo completo, **cualquier miembro del equipo** con los conocimientos y competencias necesarios puede realizar cualquier tarea, y **todos son responsables de la calidad**.

Los miembros del equipo comparten el **mismo espacio de trabajo** (físico o virtual), ya que la ubicación común facilita la comunicación y la interacción.

El enfoque de equipo completo mejora la dinámica de equipo, potencia la comunicación y la colaboración dentro del equipo y crea sinergia al permitir que los diversos conjuntos de competencias dentro del equipo se aprovechen en beneficio del proyecto.

I – Fundamentos del Proceso de Prueba

1.5.2 Enfoque de Equipo Completo

Los probadores trabajan en **estrecha colaboración con otros miembros del equipo** para asegurar que se alcanzan los niveles de calidad deseados.

Esto incluye colaborar con

- los **representantes de negocio** para ayudarles a crear pruebas de aceptación adecuadas y
- trabajar con los **desarrolladores** para acordar la estrategia de prueba y decidir los enfoques de automatización de la prueba.

De este modo, los probadores pueden transferir conocimientos sobre la prueba a otros miembros del equipo e influir en el desarrollo del producto.

Dependiendo del contexto, el enfoque de equipo completo **puede no ser siempre apropiado**.

- Por ejemplo, en algunas situaciones, como las críticas para la seguridad, puede necesitarse un alto nivel de independencia en la prueba.

1.5.3 Independencia de la Prueba

Un cierto grado de independencia hace que el probador sea **más eficaz** a la hora de encontrar defectos debido a las **diferencias entre los sesgos cognitivos del autor y del probador** (cf. Salman 1995).

Sin embargo, la independencia **no sustituye a la familiaridad**;

- por ejemplo, los desarrolladores pueden encontrar eficazmente muchos defectos en su propio código.

1.5.3 Independencia de la Prueba

Los productos del trabajo pueden ser probados por

- su autor (sin independencia),
- por los compañeros del autor del mismo equipo (cierta independencia),
- por probadores ajenos al equipo del autor pero dentro de la organización (alta independencia), o
- por probadores ajenos a la organización (independencia muy alta).

Para la mayoría de los proyectos, suele ser mejor llevar a cabo las pruebas con **varios niveles de independencia**

- por ejemplo,
  - que los desarrolladores realicen las pruebas de integración de componentes,
  - que el equipo de pruebas realice las pruebas de integración de sistemas y sistemas, y
  - que los representantes de negocio realicen las pruebas de aceptación.

1.5.3 Independencia de la Prueba

La principal **ventaja de la independencia** de la prueba es que es probable que los probadores independientes reconozcan diferentes tipos de fallos y defectos en comparación con los desarrolladores debido a sus diferentes antecedentes, perspectivas técnicas y sesgos.

Además, un probador independiente puede verificar, cuestionar o refutar las suposiciones realizadas por los implicados durante la especificación e implementación del sistema.

Sin embargo, también existen algunos **inconvenientes**.

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

I – Fundamentos del Proceso de Prueba
1.5.3 Independencia de la Prueba

Ejemplo OA FL-1.5.3:

- En series y películas vemos situaciones cómo:
  - A un policía no le permiten trabajar en un caso donde está implicado un familiar o amigo
  - A un médico no le permiten atender a un familiar o amigo
- El motivo es que las personas solemos perder cierto grado de profesionalidad ante situaciones que nos afectan de cerca
- ¿Qué crees que pasaría si se obtiene el permiso para conducir un vehículo "cuando la persona cree que está capacitado para conducir" en lugar de realizar un proceso "formal" de examen?

Organización de la Prueba. Información adicional

- Las pruebas sistemáticas de software y la gestión de la calidad disminuyen a largo plazo los costes de desarrollo y mantenimiento.
- La utilización temprana de la gestión de pruebas minimiza los riesgos en la implantación y permite acortar los tiempos del desarrollo del proyecto.
- La gestión de pruebas proporciona transparencia y es, al mismo tiempo, un indicativo valioso para el estado global del proyecto.
- Dentro del ciclo de vida de un producto deben llevarse a cabo diferentes tareas de pruebas.
- La gestión de pruebas temprana y profesional asegura a largo plazo las inversiones.

Posibles formas de llevar a cabo las pruebas. Información adicional

Pruebas del desarrollador:

- El desarrollador **nunca se mantiene neutral** respecto de su "creación".
  - En definitiva, conoce el objeto de prueba mejor que nadie.
  - No se producen costes adicionales por tener que ponerse al corriente respecto del objeto de prueba.
- El hombre **tiende a ser ciego** respecto a sus errores.
  - Por eso existe el peligro para el desarrollador de no ver fallos evidentes.
- Los errores como consecuencia de una **mala interpretación** de los requisitos quedan sin descubrir.
  - Estos errores se pueden evitar, o al menos reducir, mediante la formación de equipos de desarrollo en los que los desarrolladores comprueban mutuamente sus productos.

Pruebas diseñadas por otras personas del equipo de desarrollo.

- La formación de equipos de pruebas con miembros de diferentes áreas del proyecto mejora la calidad de las pruebas.
- Es importante que estos equipos puedan trabajar lo más independientemente posible.

1.5.3 Independencia de la Prueba

Posibles formas de llevar a cabo las pruebas. Información adicional.

Pruebas diseñadas por personas de un grupo distinto

- perteneciente a la organización (como un equipo de prueba independiente)
- especialistas en pruebas (como los especialistas en pruebas de usabilidad o prestaciones)

Pruebas diseñadas por personas de otras organizaciones

- como Outsourcing de pruebas o certificación realizada por un ente externo
- Una separación completa de las pruebas consigue la mayor distancia posible entre el objeto de prueba y el probador.
- Pero se dispone de poco conocimiento sobre el objeto de prueba y el proyecto.
  - Se requiere un gran esfuerzo de puesta al corriente.
  - Los expertos externos deberían ser por ello involucrados desde las fases tempranas del proyecto.
- Los expertos externos tienen un Know-how muy desarrollado sobre pruebas
  - La definición optima de los casos de prueba queda así garantizada.
  - Como otra ventaja, la utilización óptima de métodos y herramientas, además de una ejecución de las pruebas acorde.

I – Fundamentos del Proceso de Prueba

1.5.3 Independencia de la Prueba

Las dificultades. Información adicional.

- El probador experimentado mantiene una distancia suficiente respecto del objeto de la prueba.
  - Cuanto más alejado esté el probador del desarrollo más independientemente y más objetivamente podrá llevar a cabo su trabajo.
  - Una distancia demasiado grande respecto del objeto de prueba puede, sin embargo, provocar que se necesite más tiempo para las pruebas.
- Junto al enfoque de pruebas de aplicaciones mostrado aquí se debe tener en cuenta durante la organización de los equipos de pruebas, que también se requiere una organización de pruebas adecuada para las revisiones.

1.5.3 Independencia de la Prueba

Diferencias entre diseñar, desarrollar y probar. Información adicional

- Las pruebas requieren una forma diferente de pensar de las de aquellos que diseñan o desarrollan sistemas:
  - La misión del diseñador es ayudar al cliente suministrándole los requisitos correctos.
  - La misión del desarrollador es convertir los requisitos en funciones.
  - La misión del probador es examinar la correcta implementación de los requisitos del usuario.
  - Objetivo común: proporcionar buen software.
- En principio, **una persona puede asumir los tres roles**:
  - **Es difícil**, pero es posible.
  - Se deben tener en cuenta las diferencias en los objetivos asociados a cada rol.
  - Otras soluciones (como tener un equipo de pruebas independiente) suelen ser más fáciles y producir mejores resultados.

Final del capítulo

Actividades propuestas

- Debate:
  - ¿Coinciden los conceptos y terminología del proceso de prueba con los que usas en tu actividad profesional?
  - ¿Ves alguna oportunidad de mejora en tu proceso de prueba?
- Resumen. Trata de sintetizar en menos de una página los conceptos clave, así como los que más dificultades te han generado
- Consulta el glosario para asegurarte que cada término utilizado lo has asimilado
- Realiza ejercicios de ejemplo de examen que corresponden a este capítulo
- Realiza consultas tanto para enfrentarte adecuadamente al examen como para aplicar los conocimientos a tu proceso de prueba

Capítulo II – Prueba a lo largo del Ciclo de Vida de Desarrollo de Software (CVDS)

- II/01 La Prueba en el Contexto de un Ciclo de Vida de Desarrollo de Software
- II/02 Niveles de Prueba y Tipos de Prueba
- II/03 Pruebas de Mantenimiento

II – Prueba a lo largo del CVDS

Glosario / Palabras clave en Español e Inglés

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

II – Prueba a lo largo del CVDS

Objetivos de Aprendizaje

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

- FL-2.1.1 (K2) Explicar el impacto del ciclo de vida de desarrollo de software elegido en la prueba.
- FL-2.1.2 (K1) Recordar las buenas prácticas de prueba que se aplican a todos los ciclos de vida de desarrollo del software.
- FL-2.1.3 (K1) Recordar los ejemplos de enfoques de prueba primero para el desarrollo.
- FL-2.1.4 (K2) Resumir cómo DevOps puede tener un impacto en la prueba.
- FL-2.1.5 (K2) Explicar el enfoque de desplazamiento a la izquierda.
- FL-2.1.6 (K2) Explicar cómo se pueden utilizar las retrospectivas como mecanismo para la mejora del proceso.

II – Prueba a lo largo del CVDS

Objetivos de Aprendizaje

2.2 Niveles de Prueba y Tipos de Prueba

- FL-2.2.1 (K2) Distinguir los diferentes niveles de prueba.
- FL-2.2.2 (K2) Distinguir los diferentes tipos de prueba.
- FL-2.2.3 (K2) Distinguir la prueba de confirmación de la prueba de regresión..

2.3 Prueba de Mantenimiento

- FL-2.3.1 (K2) Resumir la prueba de mantenimiento y sus desencadenantes.

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Un modelo de ciclo de vida de desarrollo de software (**CVDS**) es una representación abstracta y de alto nivel del proceso de desarrollo de software.

Un modelo CVDS define cómo se relacionan entre sí, tanto lógica como cronológicamente, las diferentes fases de desarrollo y los tipos de actividades que se realizan dentro de este proceso.

Algunos ejemplos de modelos de CVDS son:

- los modelos de **desarrollo secuencial** (por ejemplo, el modelo en cascada, el modelo V),
- los modelos de **desarrollo iterativo** (por ejemplo, el modelo en espiral, el prototipado) y
- los modelos de **desarrollo incremental** (por ejemplo, el Proceso Unificado).

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Algunas actividades de los procesos de desarrollo de software también pueden describirse mediante métodos de desarrollo de software más detallados y **prácticas Ágiles**.

Algunos ejemplos son:

- el desarrollo guiado por pruebas de aceptación (DGPA) - acceptance test-driven development (ATDD),
- el desarrollo guiado por el comportamiento (DGC) - behavior-driven development (BDD),
- el diseño guiado por dominios (DGD) - domain-driven design (DDD),
- la programación extrema (XP) - extreme programming,
- el desarrollo guiado por prestaciones (DGP) - feature-driven development (FDD),
- Kanban,
- Lean IT,
- Scrum y
- desarrollo guiado por pruebas (DGP) - test-driven development (TDD)

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Ejemplo OA FL-2.1.1:

- En empresas de cultura y organización desarrollada en el siglo XX, es habitual encontrase
  - Equipos de probadores organizados en área distintas a las de desarrollo
  - Una alta tasa de pruebas manuales
  - Menor utilización de herramientas
  - Empresas y equipos de trabajo con un gran número de personas
  - Métodos de trabajo secuenciales
- En empresas de cultura y organización más reciente, es habitual encontrarse
  - Probadores como parte de un equipo pequeño
  - Mayor uso de herramientas y tecnología
  - Organización del trabajo más iterativo (no necesariamente ágil)
- Esto no significa que un enfoque sea siempre mejor que el otro, pero ¿Cuáles han sido tus experiencias en cada uno de estos contextos?

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Información adicional

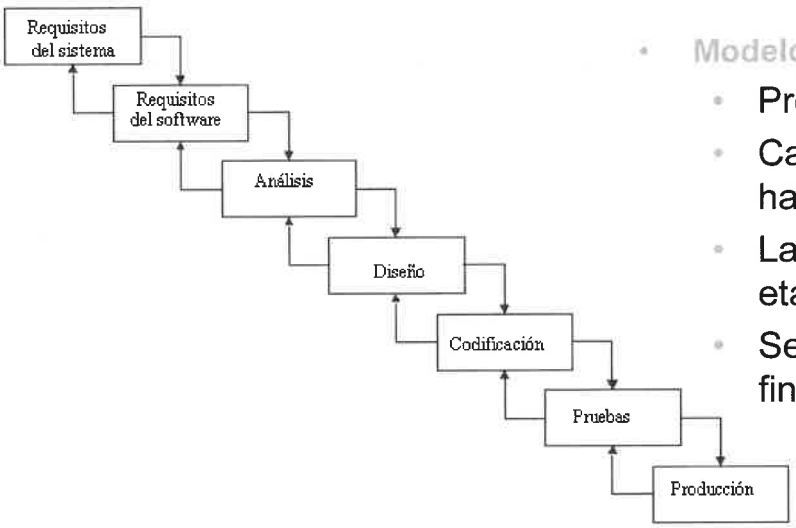
- El **modelo de desarrollo secuencial** describe el proceso como un flujo lineal y secuencial de actividades. Una fase se inicia cuando se ha completado la anterior. Aunque en la teoría no hay solapamiento de fases, en la práctica es beneficioso tener una retroalimentación temprana de la fase siguiente
  - En el **modelo en cascada**, las actividades de desarrollo se completan una tras otra. En este modelo las actividades de prueba sólo ocurren después de que las actividades de desarrollo han sido completadas
  - El **modelo en V** integra el proceso de prueba a lo largo del proceso de desarrollo, realizando prueba temprana. Este modelo incluye niveles de prueba asociados a cada fase de desarrollo, que favorece aún más la prueba temprana. Aunque cada nivel de prueba se realiza de forma secuencial, en ocasiones ocurren solapamientos
  - Los modelos secuenciales ofrecen software con un conjunto completo de prestaciones pero requieren meses o años para la entrega a los implicados y usuarios

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Información adicional

- **Un desarrollo de software estructurado (cascada) sigue pasos claramente definidos**
  - Diferentes modelos de procedimiento muestran el desarrollo estructurado



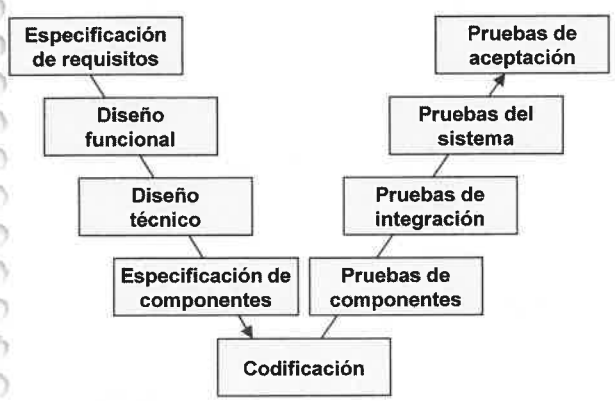
- Modelo en cascada según Royce (1970):
  - Proceso de desarrollo en 7 etapas.
  - Cada etapa sólo tiene una conexión hacia atrás con la etapa anterior.
  - Las pruebas sólo representan una etapa en el proceso .
  - Se corresponde con la aceptación final de producto.

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Información adicional. Modelo genérico en V

- El modelo genérico en V es un modelo de procedimiento para el proceso de desarrollo de software (secuencial).
- El desarrollo y las pruebas se representan mediante dos ramas de la misma consideración.



- Cada fase del desarrollo se contrapone a una fase diferente de las pruebas.
- La preparación de las pruebas se lleva a cabo de manera paralela al desarrollo de software, de hecho, en cada etapa .
- Se prueba durante el ciclo de vida completo del software.
- Puede haber más o menos fases: pruebas de integracion de componentes y de integración de sistemas.

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Información adicional

- El **modelo de desarrollo incremental** implica establecer requisitos, diseñar, construir y probar un sistema en fragmentos.
  - Las prestaciones del sistema crecen de forma incremental
  - En este modelo se especifican, diseñan, construyen y prueban conjuntamente grupos de prestaciones en base a **ciclos**, a menudo de **duración fija**.
  - Las iteraciones pueden implicar cambios en prestaciones que se desarrollaron con anterioridad, así como, cambios en el propio alcance del proyecto
  - Cada iteración proporciona software operativo, que es un subconjunto creciente del conjunto total de prestaciones.
  - El proceso se repite hasta llegar a la entrega final de todas las prestaciones o el proyecto se detiene con un alcance menor

https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Información adicional. Ejemplos de modelos de **desarrollo iterativo**.

- **Rational Unified Process**: Iteraciones relativamente largas (p.ej., de dos a tres meses), e incrementos de las prestaciones proporcionalmente grandes, como por ejemplo dos o tres grupos de prestaciones relacionadas.
- **Scrum**: Iteraciones relativamente cortas (p.ej., horas, días o unas pocas semanas), e incrementos de las prestaciones proporcionalmente pequeños, como unas pocas mejoras y/o dos o tres prestaciones nuevas.
- **Kanban**: Implementado con o sin iteraciones de longitud fija, que puede ofrecer una sola mejora o prestación una vez finalizada, o puede agrupar prestaciones para liberarlas de una sola vez.
- **Espiral** (o **prototipado**): Implica la creación de incrementos experimentales, algunos de los cuales pueden ser reelaborados en profundidad o incluso abandonados en trabajos de desarrollo posteriores.
- Rapid Application Development (RAD): p.ej. interfaz del usuario implementada mediante funcionalidad comercial, ignorando la funcionalidad de desarrollo posterior
- Extreme Programming (XP): el desarrollo y las pruebas se llevan a cabo sin que exista una especificación de requisitos formalizada.

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Información adicional

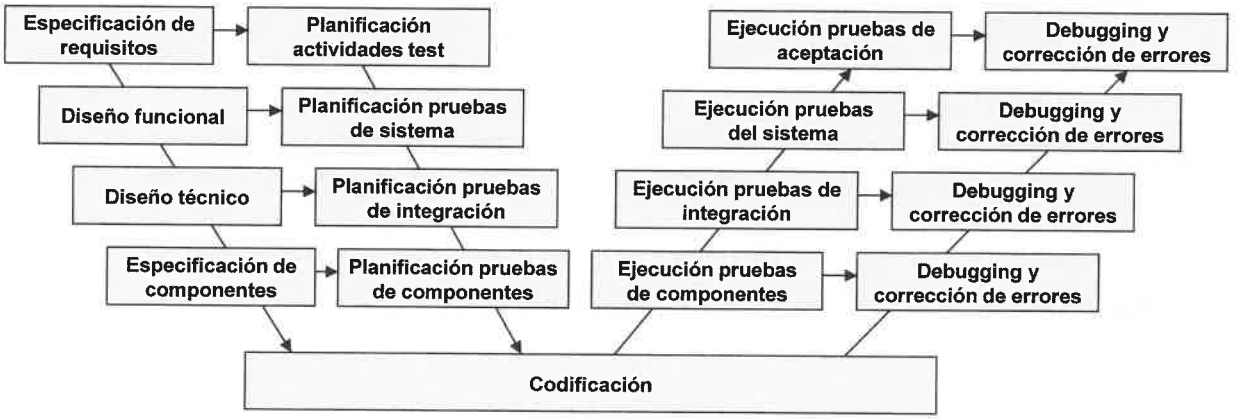
- Los componentes o sistemas desarrollados con **métodos iterativos**, implican **niveles de prueba que se solapan** a lo largo del desarrollo
- Lo ideal es que cada prestación se pruebe en varios niveles de prueba conforme se aproxime la entrega.
- En ocasiones los equipos utilizan **entrega continua** o **despliegue continuo** lo que implica una alta automatización de los niveles de prueba
- Las actividades de desarrollo incluyen el concepto de **equipos auto-organizados** que pueden cambiar la forma en la que se organiza el trabajo de prueba entre probadores y desarrolladores
- Estos métodos generan sistemas en crecimiento que se libera a usuarios finales por prestaciones, por iteraciones o a gran escala (tradicionalmente)
- La **prueba de regresión** adquiere más importación según crece el sistema
- Los modelos iterativos incrementales pueden ofrecer software utilizable en semanas (o días). Pero sólo pueden ofrecer el conjunto completo de funcionalidades en un periodo de meses (quizás años)

II – Prueba a lo largo del CVDS

2.1 La Prueba en el Contexto de un CVDS

Información adicional. Las pruebas en el modelo genérico en W.

- El modelo en W puede verse como una ampliación del modelo genérico en V
- En el modelo de W queda claro que ciertas actividades de pruebas pueden establecerse como procesos paralelos al propio desarrollo de software.



II – Prueba a lo largo del CVDS

2.1.1 Impacto del CVDS en la Prueba

La prueba debe adaptarse al CVDS para tener éxito.

La elección del CVDS repercute en:

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

II – Prueba a lo largo del CVDS

2.1.1 Impacto del CVDS en la Prueba

En las fases iniciales de los **modelos de desarrollo secuencial**,

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

En algunos **modelos de desarrollo iterativo e incremental**,

- se supone que cada iteración entrega un prototipo funcional o un incremento del producto.
- Esto implica que en cada iteración se puede probar tanto estática como dinámicamente en todos los niveles de prueba.
- La entrega frecuente de incrementos requiere una retroalimentación rápida y **pruebas de regresión extensivas**.

II – Prueba a lo largo del CVDS

2.1.1 Impacto del CVDS en la Prueba

El **desarrollo de software Ágil** asume que pueden producirse **cambios** a lo largo de todo el proyecto.

Por lo tanto, en los proyectos ágiles

- se favorece una **documentación** de producto de trabajo **ligera** y
- una **amplia automatización de la prueba** para facilitar la prueba de regresión.
- Además, la mayor parte de las pruebas manuales suelen realizarse mediante técnicas de **prueba basadas en la experiencia** (véase la sección 4.4) que no requieren un análisis y diseño de prueba previos exhaustivos.

II – Prueba a lo largo del CVDS

2.1.2 CVDS y Buenas Prácticas de Prueba

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

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

2.1.3 La Prueba como Impulsor del Desarrollo de Software

Enfoques de desarrollo, en los que las pruebas se definen como un medio para conducir el desarrollo.

- desarrollo guiado por prueba (DGP / TDD),
- desarrollo guiado por prueba de aceptación (DGPA / ATDD) y
- desarrollo guiado por el comportamiento (DGC / BDD)

Cada uno de estos enfoques aplica el principio de la **prueba temprana** (véase el apartado 1.3) y sigue un planteamiento de **desplazamiento a la izquierda** (véase el apartado 2.1.5), ya que **las pruebas se definen antes de escribir el código**.

Apoyan un modelo de desarrollo iterativo.

II – Prueba a lo largo del CVDS

2.1.3 La Prueba como Impulsor del Desarrollo de Software

Estos enfoques se caracterizan por lo siguiente (1/2):

- Desarrollo Guiado por Prueba (DGP):
  - Dirige la codificación mediante casos de prueba (en lugar de un diseño extensivo del software) (Beck 2003).
  - Primero se redactan las pruebas, luego se elabora el código para que satisfaga las pruebas y, por último, se refactorizan las pruebas y el código.
- Desarrollo Guiado por Prueba de Aceptación (DGPA) (véase el apartado 4.5.3):
  - Obtiene pruebas a partir de criterios de aceptación como parte del proceso de diseño del sistema (Gärtner 2011).
  - Las pruebas se redactan antes de que la parte de la aplicación se desarrolle para satisfacer las pruebas.

II – Prueba a lo largo del CVDS

2.1.3 La Prueba como Impulsor del Desarrollo de Software

Estos enfoques se caracterizan por lo siguiente (2/2):

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

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

II – Prueba a lo largo del CVDS

2.1.4 DevOps y la Prueba

DevOps es un enfoque organizativo que pretende crear sinergias haciendo que el desarrollo (incluidas las pruebas) y las operaciones **trabajen juntos** para alcanzar un conjunto de **objetivos comunes**.

DevOps requiere un **cambio cultural** dentro de una organización para salvar las distancias entre el desarrollo (incluidas las pruebas) y las operaciones, tratando sus funciones con el mismo valor.

DevOps promueve

- la autonomía de los equipos,
- la retroalimentación rápida,
- las cadenas de herramientas integradas y
- prácticas técnicas como la integración continua (IC) y la entrega continua (EC).

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

2.1.4 DevOps y la Prueba

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

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

II – Prueba a lo largo del CVDS

2.1.4 DevOps y la Prueba

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

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

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

2.1.4 DevOps y la Prueba

Ejemplo OA FL-2.1.4:

- Una simple búsqueda por el documento "State of DevOps Report" (se publica uno por año) nos permite tener acceso a datos que indican el impacto de la DevOps en el desarrollo de producto y, obviamente, en las pruebas
- Una entidad que sirve como referencia para conocer la cultura DevOps es https://www.dasa.org/blog/
- Aunque la fecha oficial de aparición de DevOps es 2008, la mayor parte de los valores, principios y prácticas que aplica existían con anterioridad. Por ejemplo:
  - El feedback (retroalimentación) temprano descrito en DevOps tiene en fuerte vínculo con las pruebas tempranas (o desplazamiento a la izquierda) que las propuestas de pruebas (cómo ISTQB) describían años antes
  - Entender las pruebas y la calidad como un aspecto que es de todos
- ¿Cuáles han sido tus experiencias en el contexto DevOps?

2.1.5 Enfoque "Desplazamiento a la Izquierda"

El principio de prueba temprana (véase la sección 1.3) se denomina a veces desplazamiento a la izquierda porque es un enfoque en el que la prueba se realiza de forma temprana en el CVDS.

El desplazamiento a la izquierda normalmente sugiere que la prueba debería realizarse antes (por ejemplo, no esperar a que se implemente el código o a que se integren los componentes), pero no significa que deba descuidarse la prueba más avanzada en el CVDS.

2.1.5 Enfoque "Desplazamiento a la Izquierda"

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

- **Revisión de la especificación desde la perspectiva de la prueba**. Estas actividades de revisión de las especificaciones suelen encontrar defectos potenciales, como ambigüedades, incompletitud e inconsistencias.
- **Redactar casos de prueba desde antes de escribir el código** y hacer que éste se ejecute en un arnés de prueba durante la implementación de código.
- **Utilizar integración continua** (IC) e incluso mejor **entrega continua** (EC), ya que aporta una retroalimentación rápida y pruebas de componentes automatizadas para acompañar al código fuente cuando se envía al repositorio de código.
- Completar el análisis **estático del código fuente antes de la prueba dinámica**, o como parte de un proceso automatizado.

II – Prueba a lo largo del CVDS

2.1.5 Enfoque "Desplazamiento a la Izquierda"

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

- **Realizar pruebas no funcionales empezando en el nivel de prueba de componente**, siempre que sea posible.
  - Se trata de una forma de desplazamiento a la izquierda, ya que estos tipos de pruebas no funcionales tienden a realizarse más adelante en el CVDS, cuando se dispone de un sistema completo y de un entorno de prueba representativo.

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

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

II – Prueba a lo largo del CVDS

2.1.5 Enfoque "Desplazamiento a la Izquierda"

Ejemplo y debate OA FL-2.1.5:

- Vamos a realizar un análisis grupal de las consecuencias de realizar una actividad de prueba en dos escenarios distintos:
  - Actividad de prueba realizada en momentos finales de CVDS
  - Esa misma actividad realizada en otros instantes "tempranos" del CVDS
- ¿Qué consecuencias identificas para el proyecto?

II – Prueba a lo largo del CVDS

2.1.6 Retrospectivas y Mejora de Proceso

Las **retrospectivas** (también conocidas como **"reuniones post proyecto"** y retrospectivas de proyecto) suelen celebrarse al final de un proyecto o de una iteración, en un hito de entrega, o pueden celebrarse cuando se necesiten.

La cronología y la organización de las retrospectivas dependen del modelo de CVDS concreto que se siga.

En estas reuniones los participantes (no sólo probadores, sino también, por ejemplo, desarrolladores, arquitectos, propietario de producto, analistas de negocio) **debaten**:

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

Los resultados deben registrarse y normalmente forman parte del informe de compleción de la prueba (véase la sección 5.3.2).

2.1.6 Retrospectivas y Mejora de Proceso

Las retrospectivas son fundamentales para el éxito de la implementación de la **mejora continua** y es importante que se haga un seguimiento de cualquier mejora recomendada.

Entre los **beneficios** habituales para la prueba se incluyen:

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

II – Prueba a lo largo del CVDS

2.1.6 Retrospectivas y Mejora de Proceso

Ejemplo OA FL-2.1.6:

- Vamos a realizar la siguiente actividad:
  - Identifica alguna dificultad relevante ocurrida durante el desarrollo y/o prueba de un producto que puedas comentar en este grupo
  - El grupo te pedirá información sobre lo ocurrido tratando de identificar entre todos la "causa raíz" de lo sucedido. No se trata de encontrar culpables, sólo las situaciones que lo provocaron
  - El grupo realizará propuestas de posibles formas de evitar la situación que provoca el incidente no deseado
  - Aplica sólo aquellas propuestas de mejora que veas viables por tiempo, coste, conocimiento, medios disponibles, …

II – Prueba a lo largo del CVDS

2.2 Niveles de Prueba y Tipos de Prueba

Los niveles de prueba son grupos de actividades de prueba que se organizan y gestionan conjuntamente.

- Cada nivel de prueba es una **instancia** del proceso de prueba, realizado en relación con el software en una etapa determinada de desarrollo, desde componentes individuales hasta sistemas completos o, en su caso, sistemas de sistemas.
- Los niveles de prueba están relacionados con otras actividades dentro del CVDS.
- En los modelos secuenciales de CVDS, los niveles de prueba suelen definirse de tal forma que los criterios de salida de un nivel forman parte de los criterios de entrada del siguiente.
- En algunos modelos iterativos, esto puede no aplicarse.
- Las actividades de desarrollo pueden abarcar varios niveles de prueba.
- Los niveles de prueba pueden solaparse en el tiempo.

II – Prueba a lo largo del CVDS

2.2 Niveles de Prueba y Tipos de Prueba

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

II – Prueba a lo largo del CVDS

2.2.1 Niveles de Prueba

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

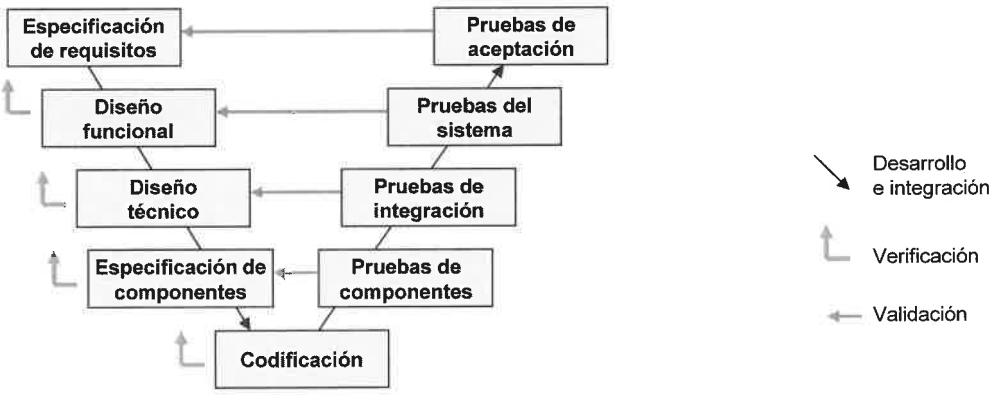
- Prueba de Componente
- Prueba de Integración de Componentes
- Prueba de Sistema
- Prueba de Integración de Sistemas
- Prueba de Aceptación

II – Prueba a lo largo del CVDS

2.2 Niveles de Prueba

Las pruebas en el modelo genérico en V. Información adicional

- Ambas partes del modelo han de ser consideradas de igual valor.
- Algunas actividades se pueden desarrollar en paralelo, de esta forma se pueden concluir con anticipación la especificación de casos de prueba / la planificación de las pruebas. Otras actividades deben ser postergadas.



II – Prueba a lo largo del CVDS

2.2.1 Niveles de Prueba

Prueba de Componente

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

II – Prueba a lo largo del CVDS

2.2.1 Niveles de Prueba

Prueba de Integración de Componentes

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

II – Prueba a lo largo del CVDS

2.2.1 Niveles de Prueba

Prueba de Sistema

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

II – Prueba a lo largo del CVDS

2.2.1 Niveles de Prueba

Prueba de Integración de Sistemas

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

2.2.1 Niveles de Prueba

Prueba de Aceptación

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

II – Prueba a lo largo del CVDS

2.2.1 Niveles de Prueba

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

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

Nota: en los anexos de la presente documentación puedes encontrar mayor detalle sobre los niveles de prueba. Aunque los contenidos de los anexos no están en el alcance del programa de estudio y del examen de certificación, es un contenido interesante para aplicarlo a tus proyectos.

II – Prueba a lo largo del CVDS

2.2.1 Niveles de Prueba

Ejemplo OA FL-2.2.1:

- Nota importante: Es habitual que algunas empresas utilicen una terminología distinta a la que hemos visto para los niveles de prueba. En caso de duda, expón al grupo la actividad que haces, qué nombre le das y vamos a tratar de clasificarla según las descripciones vistas.
- La utilización de herramientas denominadas de "pruebas unitarias" (JUnit, Nunit, etc.) sería el ejemplo más habitual de pruebas de componente y de pruebas de integración de componentes.
  - La forma de distinguir la prueba de componente de la de integración de componente es principalmente el objetivo de la prueba ya que los entornos, herramientas y técnicas suelen ser altamente similares
- Las pruebas que realizan los probadores en entornos preproductivos pueden ser el ejemplo habitual de pruebas de sistema
- Las actividades que realizan los usuarios en entornos que se suelen llamar de formación (o también los preproductivos) para comprobar que el software es adecuado a sus necesidades, suelen ser un ejemplo habitual de pruebas de aceptación

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Existen muchos tipos de prueba que pueden aplicarse en los proyectos.

En esta formación se abordan los siguientes cuatro tipos de prueba:

- Prueba Funcional
- Prueba No Funcional
- Prueba de Caja Negra
- Prueba de Caja Blanca

Los cuatro tipos de prueba mencionados pueden aplicarse a todos los niveles de prueba, aunque el enfoque será diferente en cada nivel.

Se pueden utilizar diferentes técnicas de prueba para obtener condiciones y casos de prueba para todos los tipos de prueba mencionados.

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba Funcional

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

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba No Funcional

- La prueba no funcional evalúa atributos distintos de las características funcionales de un componente o sistema.
- La prueba no funcional es la comprobación de "lo bien que se comporta el sistema".
- El principal objetivo de la prueba no funcional es comprobar las **características de calidad del software no funcionales**.

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba No Funcional

El estándar ISO/IEC 25010 proporciona la siguiente clasificación de las características de calidad del software no funcionales:

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

Consulte el programa de estudios de ISTQB Analista de Pruebas Técnicas si desea información adicional

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba No Funcional

A veces es conveniente que **la prueba no funcional comience al principio del ciclo de vida** (por ejemplo, como parte de las revisiones y prueba de componente o prueba de sistema).

- Muchas pruebas no funcionales se obtienen a partir de pruebas funcionales, ya que utilizan las mismas pruebas funcionales, pero comprueban que, al realizar la función, se satisface una restricción no funcional
  - por ejemplo, comprobar que una función se lleva a cabo en un tiempo determinado o que una función puede portarse a una nueva plataforma.
- **El descubrimiento tardío de defectos no funcionales puede suponer una grave amenaza para el éxito de un proyecto**.
- A veces, la prueba no funcional necesita un **entorno de prueba muy específico**, como un laboratorio de usabilidad para probar la usabilidad.

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba de Caja Negra

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

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba de Caja Blanca

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

2.2.2 Tipos de Prueba

Ejemplo OA FL-2.2.2

- Nota importante: Al igual que en los niveles de prueba, es habitual que algunas empresas utilicen una terminología distinta a la que hemos visto para los tipos de prueba. Expón cualquier inquietud.
- Pruebas funcionales:
  - Una gran parte de las comprobaciones de los probadores sobre las acciones que debe realizar el sistema, serían un buen ejemplo de pruebas funciones a nivel de sistema
  - También lo que hace el equipo de desarrollo cuando comprueba, por ejemplo, un cálculo que realiza un módulo sería una prueba funcional, pero a nivel de componente
- Pruebas no funcionales
  - Un ejemplo muy conocido son las pruebas para conocer los tiempos de respuesta del software con herramientas como Jmeter
  - Pero analizar la seguridad del sistema frente a accesos no autorizados sería una prueba no funcional, en este caso de seguridad

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Ejemplo OA FL-2.2.2

- Pruebas de caja negra
  - Cómo se verá en el capitulo 4, cuando comprobamos un software (sea un sistema o un módulo del mismo) sin importarnos cómo está construido sería un ejemplo habitual de caja negra. Esta comprobación puede ser de una funcionalidad (como un cálculo) o de una característica como el tiempo de respuesta.
- Pruebas de caja blanca
  - Cuando examinamos el software (sistema o módulo) viendo cómo está construido y los posibles incidentes que pueden suceder, sería un ejemplo habitual de caja blanca. La situación más típica es un programador examinando su código para detectar posibles comportamientos no deseados como una asignación incorrecta del valor de una variable o la ejecución de un código que no se corresponde al esperado al evaluar incorrectamente sobre un valor (una decisión implementada con una sentencia IF)

II – Prueba a lo largo del CVDS

2.2.3 Prueba de Confirmación y Prueba de Regresión

En general los **cambios** que se realizan en un componente o sistema son para mejorarlo añadiendo una nueva prestación o para arreglarlo eliminando un defecto.

La prueba debería entonces incluir también

- la **prueba de confirmación** y
- la **prueba de regresión**.

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

II – Prueba a lo largo del CVDS

2.2.3 Prueba de Confirmación y Prueba de Regresión

La prueba de confirmación confirma que un defecto original se ha corregido con éxito.

En función del riesgo, se puede probar la versión corregida del software de varias maneras, entre ellas:

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

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

2.2.3 Prueba de Confirmación y Prueba de Regresión

La **prueba de regresión** confirma que no se han producido consecuencias adversas a causa de un cambio, incluida una corrección que ya ha sido sometida a una prueba de confirmación.

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

II – Prueba a lo largo del CVDS

2.2.3 Prueba de Confirmación y Prueba de Regresión

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

Nota: En los anexos se amplía la información sobre los tipos de prueba, aunque fuera del alcance del programa de estudios

II – Prueba a lo largo del CVDS

2.2.3 Prueba de Confirmación y Prueba de Regresión

Ciclos de vida iterativos e incrementales. Información adicional

- Especialmente en los ciclos de vida de desarrollo iterativos e incrementales (por ejemplo, Agile), las nuevas características, los cambios en las características existentes y la refactorización del código dan como resultado **cambios frecuentes en el código**, lo que también requiere pruebas asociadas al cambio.
- Debido a la naturaleza evolutiva del sistema, la prueba de confirmación y la prueba regresión son muy importantes.

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Ejemplo OA FL-2.2.3

- Una situación muy habitual que se corresponde con una prueba de confirmación es la repetición de un caso de prueba que falló por parte de un probador después de que el equipo de desarrollo le comunica que ha corregido el defecto
- Un ejemplo de prueba de regresión en un sistema bancario sería comprobar que una trasferencia normal cuya funcionalidad no ha cambiado sigue funcionando correctamente después de actualizar el envío de dinero con Bizum a pesar de pertenecer a módulos distintos

Existen diferentes categorías de mantenimiento, puede ser

- correctivo,
- de adaptación (o adaptativo) a cambios en el entorno o
- para mejorar el rendimiento o la mantenibilidad (véase ISO/IEC 14764 para más detalles),

por lo que el mantenimiento puede implicar

- entregas/despliegues planificados y
- entregas/despliegues no planificados (correcciones en caliente)."

II – Prueba a lo largo del CVDS
2.3 Prueba de Mantenimiento

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

Probar los cambios de un sistema en producción incluye

- tanto la evaluación del éxito de la implementación del **cambio**
- como la comprobación de posibles **regresiones** en las partes del sistema que permanecen inalteradas (que suele ser la mayor parte del sistema).

El **alcance de las pruebas de mantenimiento** suele depender de:

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

II – Prueba a lo largo del CVDS

2.3 Prueba de Mantenimiento

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

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

II – Prueba a lo largo del CVDS

2.3 Prueba de Mantenimiento

Prueba de Mantenimiento. Información adicional

- Pruebas después de la aceptación. El cliente ha aceptado el producto y lo ha llevado a producción
  - Las fases propias del desarrollo y las pruebas relacionadas han finalizado.
- El software mismo está sin embargo al comienzo de su vida útil
  - A menudo se implanta para muchos años y se sigue desarrollando.
  - Seguro que sigue conteniendo errores y por ello se sigue trabajando sobre él.
  - Debe ser adaptado a nuevas condiciones o integrado en un nuevo entorno.
- ¡Cada nueva versión del producto, cada nueva actualización y cualquier otro tipo de cambio debe ser probado de nuevo!
- Las pruebas deberían ser **repetibles** tanto si van a utilizarse como prueba nueva o prueba de confirmación, con el fin de ayudar a las pruebas de regresión

Prueba de Mantenimiento. Información adicional

- Mantenimiento de software (**Correctivo** y **adaptativo**)
  - El mantenimiento de software distingue dos tipos de actividades
    - El propio mantenimiento: eliminación de errores que ya existen desde el desarrollo del software (correctivo)
    - Cuidado del software: adaptación del software a condiciones de implantación modificadas (adaptativo)
  - Pruebas de las modificaciones y eventualmente errores debido a las mismas
- Continuación del desarrollo del software (**Evolutivo**)
  - La continuación del desarrollo abarca medidas que se salen del mantenimiento (continuación planificada del desarrollo)
    - Ampliación de las interfases
    - Ampliación de funciones
  - Aquí se vuelve a recorrer otra vez el modelo de desarrollo

Ejemplo OA FL-2.3.1:

- Nota importante: Al igual que en los niveles de prueba y los tipos de prueba, es habitual que usemos una terminología distinta cuando tratamos las pruebas de mantenimiento. Expón cualquier inquietud.
- La prueba de mantenimiento es en muchas situaciones una actividad muy habitual de los probadores ya que prueban un software que lleva años utilizándose y que sufre actualizaciones.
- Dado que es habitual encontrase malas prácticas durante el mantenimiento del software (y sus pruebas) es más fácil encontrar contraejemplos (cómo no debe hacerse). Contraejemplos:
  - Descubres que tus casos de prueba fallan porque el software cambió ☹
  - Te comunican que tienes 1 semana para probar los cambios, que ya están aplicados
  - Se asume que un cambio pequeño no va a ocasionar problemas
  - Retirar una aplicación no requiere ningún análisis previo
  - Todo esto **NO** debería ocurrir

II – Prueba a lo largo del CVDS

Final del capítulo

Actividades propuestas

- Debate:
  - ¿Coinciden los conceptos y terminología de la prueba a lo largo del ciclo de vida de desarrollo con los que usas en tu proyecto?
  - ¿Ves alguna oportunidad de mejora en tu ciclo de vida de desarrollo?
- Resumen. Trata de sintetizar en menos de una página los conceptos clave así como los que más dificultades te han generado
- Consulta el glosario para asegurarte que cada término utilizado lo has asimilado
- Realiza ejercicios de ejemplo de examen que corresponden a este capítulo
- Realiza consultas tanto para enfrentarte adecuadamente al examen como para aplicar los conocimientos a tu modelo de desarrollo

Capítulo III – Prueba Estática

- III/01 Prueba Estática - Fundamentos
- III/02 Retroalimentación y Proceso de Revisión

III – Prueba Estática

Glosario / Palabras clave en Español e Inglés

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

III – Prueba Estática

Objetivos de Aprendizaje

3.1 Prueba estática - Fundamentos

- FL-3.1.1 (K1) Reconocer los tipos de productos que pueden ser evaluados mediante las diferentes técnicas de prueba estática.
- FL-3.1.2 (K2) Explicar el valor de la prueba estática.
- FL-3.1.3 (K2) Comparar y contrastar la prueba estática y la prueba dinámica.

3.2 Retroalimentación y Proceso de Revisión

- FL-3.2.1 (K1) Identificar las ventajas de una retroalimentación temprana y frecuente de los implicados.
- FL-3.2.2 (K2) Resumir las actividades del proceso de revisión.
- FL-3.2.3 (K1) Recordar qué responsabilidades se asignan a los principales roles a la hora de realizar revisiones.
- FL-3.2.4 (K2) Comparar y contrastar los diferentes tipos de revisión.
- FL-3.2.5 (K1) Recordar los factores que contribuyen al éxito de una revisión.

III – Prueba Estática

3.1 Prueba Estática - Fundamentos

A diferencia de la prueba dinámica, **en la prueba estática no es necesario ejecutar el software sujeto a prueba**.

El código, la especificación del proceso, la especificación de la arquitectura del sistema u otros productos de trabajo se evalúan:

- mediante un examen manual (por ejemplo, revisiones) o
- con la ayuda de una herramienta (por ejemplo, el **análisis estático**).

Los **objetivos** de prueba incluyen

- la mejora de la calidad,
- la detección de defectos
- y la evaluación de características como la legibilidad, la completitud, la corrección, la capacidad de ser probado y la consistencia.

La prueba estática puede aplicarse tanto para la verificación como para la validación.

III – Prueba Estática

3.1.1 Productos de Trabajo Susceptibles de Ser Examinados Mediante Prueba Estática

Cualquier producto de trabajo que pueda leerse y comprenderse puede ser objeto de una revisión.

Sin embargo, para el **análisis estático**, los productos de trabajo **necesitan una estructura** con la que puedan ser comprobados (por ejemplo, modelos, código o texto con una sintaxis formal).

Los productos de trabajo que no son apropiados para las pruebas estáticas incluyen

- aquellos que son difíciles de interpretar por los seres humanos y
- que no deben ser analizados por herramientas (por ejemplo, el código ejecutable de terceros debido a razones legales).

III – Prueba Estática

3.1.2 Valor de la Prueba Estática

- La prueba estática puede **detectar defectos en las fases más tempranas** del CVDS, cumpliendo el principio de la prueba temprana (véase la sección 1.3).
- También puede **identificar defectos que no pueden detectarse mediante pruebas dinámicas** (por ejemplo, código inaccesible, patrones de diseño no implementados como se deseaba, defectos en productos de trabajo no ejecutables).
- La prueba estática permite **evaluar la calidad** de los productos de trabajo y **generar confianza** en ellos.
  - Al verificar los requisitos documentados, los implicados también pueden asegurarse de que dichos requisitos describen sus **necesidades reales**.
  - Dado que la prueba estática puede realizarse en una fase temprana del CVDS, puede crearse un **entendimiento compartido** entre los implicados.
  - También mejorará la **comunicación** entre los implicados.
  - Por este motivo, se recomienda involucrar a una **amplia variedad** de implicados en la prueba estática. (Pero no cantidad)

3.1.2 Valor de la Prueba Estática

- Aunque la implementación de **las revisiones puede resultar costosa**, los **costes totales del proyecto suelen ser mucho menores** que cuando no se realizan revisiones, ya que se necesita dedicar menos tiempo y esfuerzo a la corrección de defectos cuando el proyecto se encuentra más avanzado.
- Los defectos de código pueden detectarse utilizando el análisis estático de forma más eficiente que en la prueba dinámica, lo que suele dar lugar tanto a un menor número de defectos de código como a un **menor esfuerzo general de desarrollo**.

III – Prueba Estática

3.1.2 Valor de la Prueba Estática

Ejemplo OA FL-3.1.2:

- A pesar de que pueda parecer contraintuitivo, muchos incidentes graves con el software podrían haberse evitado con una adecuada revisión. Adicionalmente, en muchos casos, las pruebas dinámicas no pudieron evitar el incidente, sólo constatar el problema
- Suceso real: Aplicativo que se detectó anormalmente lento en las fases finales del proyecto. El incidente supuso un retraso y sobrecoste significativo al proyecto debido a la enorme dificultad para encontrar el defecto. Una revisión de un minuto de la consulta a la BBDD que ocasionó el incidente hubiera lo evitado.

III – Prueba Estática

3.1.3 Diferencias entre la Prueba Estática y la Prueba Dinámica

La **prueba estática y la prueba dinámica** son prácticas complementarias.

Tienen objetivos similares, como apoyar la detección de defectos en los productos de trabajo (véase la sección 1.1.1), pero también existen **algunas diferencias**, como: (1/2)

- Tanto la prueba estática como la dinámica (con análisis de fallos) pueden conducir a la detección de defectos; sin embargo, **hay algunos tipos de defectos que sólo pueden encontrarse mediante pruebas estáticas o dinámicas**.
- **La prueba estática encuentra los defectos directamente**, mientras que **la prueba dinámica provoca fallos** a partir de los cuales se determinan los defectos asociados mediante un análisis posterior.
- La prueba estática puede **detectar más fácilmente los defectos**
  - que se encuentran en **caminos a través del código que rara vez se ejecutan**
  - o que son **difíciles de alcanzar** mediante la prueba dinámica.

III – Prueba Estática

3.1.3 Diferencias entre la Prueba Estática y la Prueba Dinámica

algunas diferencias, como: (2/2)

- La prueba estática puede aplicarse a productos de trabajo **no ejecutables**, mientras que la prueba dinámica sólo puede aplicarse a productos de trabajo **ejecutables**.
- La **prueba estática** se puede utilizar para medir las características de calidad que **no dependen de la ejecución** del código (por ejemplo, la mantenibilidad), mientras que la **prueba dinámica** se puede utilizar para medir las características de calidad que **dependen de la ejecución** del código (por ejemplo, la eficiencia de rendimiento).

Prueba estática	Prueba dinámica
Detecta defectos que las pruebas dinámicas no pueden	Provoca fallos que podrían ser causados por un defecto
Encuentra defectos directamente	Provoca fallos que permiten encontrar el defecto
Encuentra defectos en caminos atípicos o inalcanzables	No aplica
Puede aplicarse a productos no ejecutables	Aplica a productos ejecutables, como el código y el propio ejecutable (binario)
Mide características que no dependen de la ejecución (p.e. mantenibilidad)	Mide características que dependen de la ejecución (p.e. rendimiento)

OA FL-3.1.3

III – Prueba Estática

3.1.3 Diferencias entre la Prueba Estática y la Prueba Dinámica

Entre los defectos típicos que son **más fáciles y/o más económicos** de encontrar mediante la prueba estática se incluyen: (1/2)

- **Defectos en los requisitos** (por ejemplo, inconsistencias, ambigüedades, contradicciones, omisiones, imprecisiones, duplicaciones).
- **Defectos de diseño** (por ejemplo, estructuras de bases de datos ineficaces, modularidad deficiente).
- Ciertos tipos de **defectos de codificación** (por ejemplo, variables con valores indefinidos, variables no declaradas, código inalcanzable o duplicado, excesiva complejidad del código).
- **Desviaciones de los estándares** (por ejemplo, falta de adherencia a las convenciones de nomenclatura en los estándares de codificación).

III – Prueba Estática

3.1.3 Diferencias entre la Prueba Estática y la Prueba Dinámica

Entre los defectos típicos que son **más fáciles y/o más económicos** de encontrar mediante la prueba estática se incluyen: (2/2)

- **Especificaciones incorrectas de la interfaz** (por ejemplo, número, tipo u orden de los parámetros no coincidentes).
- Tipos específicos de **vulnerabilidades de seguridad** (por ejemplo, desbordamientos de memoria intermedia).
- **Lagunas o imprecisiones** en la cobertura de la **base de prueba** (por ejemplo, falta de pruebas para un criterio de aceptación).

III – Prueba Estática

3.1.3 Diferencias entre la Prueba Estática y la Prueba Dinámica

Ejemplo OA FL-3.1.3:

- En proyectos reales es habitual encontrar estas malas prácticas:
  - No considerar las revisiones como parte de las pruebas
  - Confiar excesivamente en las pruebas dinámicas
  - No tener una conciencia de la importancia de la prueba (estática o dinámica)
- Nuevamente en más fácil encontrar contraejemplos que ejemplos
- Esta falta conocimiento y de aplicación de buenas prácticas en la utilización de pruebas estáticas y dinámicas provoca (en proyectos reales):
  - Se descubren los problemas pero difícilmente se evitan
  - El número de malentendidos por defectos en documentación es significativamente elevado
  - El proyecto se relentiza y se encarece y no se conoce la causa.
- Si tienes malas experiencias en tus proyectos que no puedes explicar, exponlas al grupo

III – Prueba Estática

3.2 Retroalimentación y Proceso de Revisión

Página intencionalmente en blanco

III – Prueba Estática

3.2.1 Beneficios de una Retroalimentación Temprana y Frecuente de los Implicados

La retroalimentación temprana y frecuente permite la **comunicación temprana** de posibles problemas de calidad.

- Si hay **poca implicación** por parte de los implicados durante el CVDS, es posible que **el producto que se desarrolle no cumpla la visión original** o actual del implicado.
- Un fallo en la entrega de lo que la parte interesada **desea** puede dar lugar a
  - una costosa repetición del trabajo,
  - incumplimiento de los plazos,
  - atribuciones de culpas
  - e incluso podría llevar al fracaso total del proyecto.

3.2.1 Beneficios de una Retroalimentación Temprana y Frecuente de los Implicados

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

3.2.2 Actividades del Proceso de Revisión

El estándar ISO/IEC 20246 define un **proceso de revisión genérico** que proporciona un marco estructurado pero flexible a partir del cual se puede adaptar un proceso de revisión específico a una situación concreta.

- Si la revisión requerida es **más formal**, entonces se necesitarán **más de las tareas** descritas para las distintas actividades.
- El tamaño de muchos productos de trabajo hace que sean **demasiado grandes** para ser cubiertos por una sola revisión.
- El proceso de revisión puede invocarse un par de veces para completar la revisión de todo el producto de trabajo. (**repetición**)

3.2.2 Actividades del Proceso de Revisión

Las actividades del proceso de revisión son:

- Planificación.
- Inicio de la Revisión.
- Revisión Individual.
- Comunicación y Análisis.
- Corrección y Suministro de Información.

III – Prueba Estática

3.2.2 Actividades del Proceso de Revisión

Planificación.

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

3.2.2 Actividades del Proceso de Revisión

Inicio de la Revisión.

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

3.2.2 Actividades del Proceso de Revisión

Revisión Individual.

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

3.2.2 Actividades del Proceso de Revisión

Comunicación y Análisis.

- Dado que **las anomalías identificadas durante una revisión no son necesariamente defectos**, todas estas anomalías necesitan ser analizadas y discutidas.
- Para cada anomalía, debe tomarse una decisión sobre su estado, propiedad y acciones requeridas.
- Esto suele hacerse en una **reunión de revisión**, durante la cual los participantes también deciden cuál es el nivel de calidad del producto del trabajo revisado y qué acciones de seguimiento se requieren.
- Puede ser necesaria una revisión de seguimiento para completar las acciones.

III – Prueba Estática

3.2.2 Actividades del Proceso de Revisión

Corrección y Suministro de Información.

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

3.2.2 Actividades del Proceso de Revisión

Ejemplo OA FL-3.2.2.

Algunos ejemplos de actividades habituales en las revisiones son (1/2):

- Planificación
  - Identificar el inicio y el fin de las actividades y su duración
  - Identificar los posibles participantes
  - Identificar los documentos a revisar
  - Por desgracia es poco habitual definir un objetivo concreto y las métricas usadas para monitorizarlo
- Inicio de revisión
  - Reunir a los participantes para explicar el objetivos y detalles de proceso
  - Resolver dudas sobre la actividad a realizar
- Revisión individual
  - Leer la documentación y realizar notas y comentarios
  - Aquí se pueden aplicar técnicas de revisión. Es sencillo encontrar documentación sobre ellas, p.e utilización de roles o escenarios

III – Prueba Estática

3.2.2 Actividades del Proceso de Revisión

Ejemplo OA FL-3.2.2.

Algunos ejemplos de actividades habituales en las revisiones son (2/2):

- Comunicación y análisis
  - Suele ser habitual una reunión para identificar todas las notas, consensuarlas y decidir los cambios a realizar. En ocasiones se comete el error de pensar que la revisión se ciñe a esta actividad
  - Dependiendo del tipo de revisión, los informes y documentación de gestión de la revisión admite varios formatos y estructuras ¿dispone tu proyecto de estas plantillas?
- Corrección y suministro de información
  - La actividad por excelencia será la corrección del documento por el autor
  - Aquí se elaboran los informes de revisión y la documentación del proceso de revisión para futuras mejoras y futuros análisis

3.2.3 Roles y Responsabilidades en las Revisiones

Las revisiones implican a varios implicados, que pueden asumir varios roles.

Los principales roles y sus responsabilidades son: (1/2)

- **Gestor** - decide qué se va a revisar y aporta recursos, como personal y tiempo para la revisión
- **Autor** - crea y corrige el producto del trabajo objeto de revisión
- **Moderador** (también conocido como **facilitador**) - asegura el funcionamiento eficaz de las reuniones de revisión, incluyendo la mediación, la gestión del tiempo y un entorno de revisión seguro en el que todos puedan hablar libremente.
- **Escriba** (también conocido como **grabador**) - recopila las anomalías de los revisores y registra la información de la revisión, como las decisiones y las nuevas anomalías encontradas durante la reunión de revisión.

III – Prueba Estática

3.2.3 Roles y Responsabilidades en las Revisiones

Los principales roles y sus responsabilidades son: (1/2)

- **Revisor** - lleva a cabo las revisiones. Un revisor puede ser alguien que trabaje en el proyecto, un experto en la materia o cualquier otro implicado
- **Líder de la revisión** - asume la responsabilidad general de la revisión, como decidir quién participará y organizar cuándo y dónde tendrá lugar la revisión

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

3.2.4 Tipos de Revisión

Existen muchos tipos de revisión, desde revisiones informales hasta revisiones formales.

- El **nivel de formalidad** requerido depende de factores como el CVDS que se siga, la madurez del proceso de desarrollo, la criticidad y complejidad del producto de trabajo que se revise, los requisitos legales o reglamentarios y la necesidad de un rastro de auditoría.
- Un mismo producto de trabajo puede ser revisado con distintos tipos de revisión, por ejemplo, primero una informal y después una más formal.
- **Seleccionar el tipo de revisión adecuado es clave** para alcanzar los objetivos de revisión requeridos (véase la sección 3.2.5).
- La selección no sólo se basa en los objetivos, sino también en factores como las necesidades del proyecto, los recursos disponibles, el tipo de producto y los riesgos del trabajo, el dominio del negocio y la cultura de la empresa.

III – Prueba Estática

3.2.4 Tipos de Revisión

Algunos de los tipos de revisión más utilizados son:

- Revisión Informal.
- Revisión Guiada.
- Revisión Técnica.
- Inspección.

III – Prueba Estática

3.2.4 Tipos de Revisión

Revisión Informal.

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

Revisión Guiada.

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

III – Prueba Estática

3.2.4 Tipos de Revisión

Revisión Informal. Información adicional.

- Objetivo: Forma barata de obtener (algún) beneficio
- A menudo iniciada por el autor
- Se organizan y desarrollan rápidamente
- Coste bajo
- No suele haber documentación

3.2.4 Tipos de Revisión

Revisión Guiada (Walkthrough). Información adicional.

- Sesiones abiertas / de duración indefinida (open-ended)
- Propósitos: aprender, ganar conocimiento y encontrar defectos
- Ejemplos para la utilización de Walkthroughs
  - Walkthrough de documentación
  - Walkthrough de propuestas para GUI
  - Walkthrough de procesos de negocio
- Menor esfuerzo en la preparación – por contra en ocasiones reuniones más larga
- La reunión se puede iniciar a corto plazo
- El autor tiene mucha influencia (precisamente como director de la reunión) – se corre el peligro de no discutir suficientemente acerca de puntos críticos.
- No hay control, ya que el autor es el encargado de la elaboración posterior

3.2.4 Tipos de Revisión

Revisión Técnica.

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

Revisión Técnica. Información adicional

- Comprobación de los aspectos técnicos del objeto de prueba
- Comprobación sólo en base a los requisitos y al patrón de pruebas
- Preparación previa a la reunión
- Puede variar desde bastante informal a muy formal
- Aumento de la comunicación
- Muy eficiente si el personal es adecuado (y los procedimientos correctos)
- Se requiere disponer de personal experto (con tiempo disponible, lo que no siempre es fácil)
- Gran esfuerzo para la preparación y la elaboración posterior
- Es posible una búsqueda de errores estructurada

III – Prueba Estática
3.2.4 Tipos de Revisión

Inspección.

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

III – Prueba Estática

3.2.4 Tipos de Revisión

Inspección. Información adicional.

- Previamente se establece la capacidad de que el objeto de prueba pueda ser revisado. El objetivo es encontrar defectos.
- Reunión que se desarrolla con una buena organización
- Gran esfuerzo para la preparación y la elaboración posterior
- Es posible una búsqueda de errores estructurada

III – Prueba Estática

3.2.4 Tipos de Revisión

Ejemplo OA FL-3.2.4.

- Quizás la situación más conocida en los proyectos reales sea la realización de revisiones informales. Las ceremonias realizadas en enfoques ágiles, como Scrum, suelen contener una revisión informal (p.e. el debate al explicar las historias por Propietario del Producto)
- Las revisiones guiadas suelen ser menos habituales en nuestros proyectos, pero cuando el autor de un documento participa activamente en la exposición de sus contenidos y toma nota de los comentarios, podemos estar aplicando este tipo de revisión
- La revisión técnica suele ser habituales en la creación de producto de empresas de ingeniería (tanto que tenga o no software)
- Las revisiones formales suelen ser habituales en empresas de creación de producto crítico. El sector farmacéutico o aeroespacial son ejemplos típicos.

Información adicional

Aplicación de Técnicas de Revisión. Información adicional

- Hay diversas **técnicas de revisión** que pueden aplicarse durante la actividad de revisión individual (es decir, la preparación individual) para detectar defectos.
- Estas técnicas se pueden utilizar en todos los tipos de revisión descritos anteriormente.
- La efectividad de las técnicas puede variar según el tipo de revisión utilizada.
- A continuación, se enumeran ejemplos de diferentes técnicas de revisión individual para diversos tipos de revisión:
  - Ad hoc
  - Basada en Lista de Comprobación
  - Escenarios y Ensayos
  - Basada en Roles
  - Basada en Perspectiva

3.2.5 Factores de Éxito de las Revisiones

Hay diversos factores que determinan el éxito de las revisiones, entre los que se incluyen: (1/2)

- La definición de **objetivos claros** y **criterios de salida medibles**. La evaluación de los participantes nunca debe ser un objetivo.
- **Elegir el tipo de revisión adecuado** para alcanzar los objetivos fijados y para adaptarse al tipo de producto de trabajo, a los participantes en la revisión, a las necesidades del proyecto y al contexto.
- **Realizar las revisiones en pequeños fragmentos**, para que los revisores no pierdan la concentración durante una revisión individual y/o la reunión de revisión (cuando se celebre).
- **Proporcionar retroalimentación** de las revisiones a los implicados y a los autores para que puedan mejorar el producto y sus actividades (véase la sección 3.2.1).

III – Prueba Estática

3.2.5 Factores de Éxito de las Revisiones

Hay diversos factores que determinan el éxito de las revisiones, entre los que se incluyen: (2/2)

- Proporcionar **tiempo suficiente** a los participantes para prepararse para la revisión.
- **Apoyo de la dirección** al proceso de revisión.
- Hacer **que las revisiones formen parte de la cultura de la organización**, para promover el aprendizaje y la mejora del proceso.
- Proporcionar una **formación adecuada** a todos los participantes para que sepan cómo desempeñar su rol.
- **Facilitar las reuniones**.

III – Prueba Estática

3.2.5 Factores de Éxito para las Revisiones

Factores de Éxito para las Revisiones. Información adicional

- Se han de identificar los documentos más importantes para el proyecto. Tiene que haber un ROI claro. NO se debe revisar todo (las revisiones son costosas)
- Hay que considerar los aspectos personales de las revisiones
  - Los defectos encontrados son bienvenidos y se expresan de forma objetiva
  - Se consideran los aspectos psicológicos y personales (por ejemplo, hacer que sean experiencias positivas para el autor)
  - Estos aspectos han de ser considerados cuando se está dando la formación previa a la revisión
- Facilitar el trabajo:
  - Ser formal cuando sea necesario. No entrar a niveles de detalle demasiado altos o demasiado teóricos si no es necesario
  - Utilizar listas de comprobación (checklists) o incorporar roles cuando ayude a incrementar la efectividad o la identificación de defectos
- Cuidar la presentación de los resultados. Debe ser clara, concisa, objetiva y no herir susceptibilidades si no es necesario

III – Prueba Estática

Final del capítulo

Actividades propuestas

- Debate:
  - ¿Coinciden los conceptos y terminología de las revisiones con los que usas en tu proyecto?
  - ¿Ves la conveniencia o necesidad de las revisiones en tu equipo?
- Resumen. Trata de sintetizar en menos de una página los conceptos clave así como los que más dificultades te han generado
- Consulta el glosario para asegurarte que cada término utilizado lo has asimilado
- Realiza ejercicios de ejemplo de examen que corresponden a este capítulo
- Realiza consultas tanto para enfrentarte adecuadamente al examen como para aplicar los conocimientos a tus revisiones

IV – Análisis y Diseño de la Prueba

Objetivos de Aprendizaje

4.3 Técnicas de Prueba de Caja Blanca

- FL-4.3.1 (K2) Explicar la prueba de sentencia.
- FL-4.3.2 (K2) Explicar la prueba de rama.
- FL-4.3.3 (K2) Explicar el valor de la prueba de caja blanca..

4.4 Técnicas de Prueba Basadas en la Experiencia

- FL-4.4.1 (K2) Explicar la predicción de errores.
- FL-4.4.2 (K2) Explicar la prueba exploratoria.
- FL-4.4.3 (K2) Explicar la prueba basada en lista de comprobación.

4.5. Enfoques de Prueba Basados en la Colaboración

- FL-4.5.1 (K2) Explicar cómo escribir historias de usuario en colaboración con desarrolladores y representantes de negocio.
- FL-4.5.2 (K2) Clasificar las diferentes opciones para escribir criterios de aceptación.
- FL-4.5.3 (K3) Utilizar el desarrollo guiado por prueba de aceptación (DGPA) para obtener casos de prueba.

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Las **técnicas de prueba** ayudan al probador en el **análisis de prueba** (qué probar) y en el **diseño de prueba** (cómo probar).

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

En esta formación, las técnicas de prueba se clasifican en

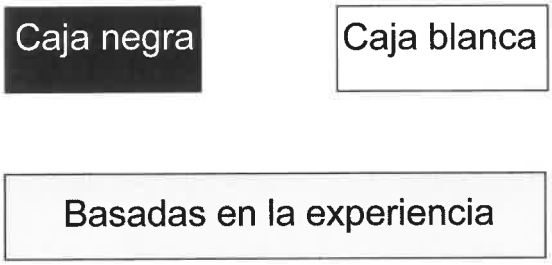
- técnicas de caja negra,
- de caja blanca y
- basadas en la experiencia.

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Generalidades

- Los procesos de prueba dinámicos se dividen en tres categorías básicas



- La división se orienta a los fundamentos respectivos para la especificación de casos de prueba
- Cada una de las categorías engloba diferentes procedimientos para la especificación de los casos de prueba para las pruebas dinámicas

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Técnicas de Prueba de Caja Negra

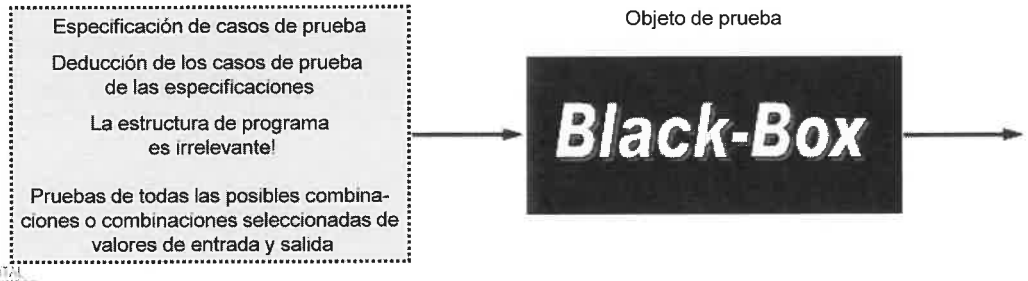
- Las técnicas de prueba de caja negra (también conocidas como **técnicas basadas en la especificación**) se basan en un análisis del comportamiento especificado del objeto de prueba sin hacer referencia a su estructura interna.
- Por lo tanto, los casos de prueba son independientes de cómo se implemente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

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Técnicas de caja negra. Información adicional.

- También llamadas técnicas conductuales o basadas en el comportamiento
- Se basan en un análisis de la base de prueba adecuada (por ejemplo, documentos de requisitos formales, especificaciones, casos de uso, historias de usuarios o procesos de negocio).
- Estas técnicas son aplicables tanto a la prueba funcional como a la no funcional.
- Las técnicas de prueba de caja negra se concentran en las entradas y salidas del objeto de prueba sin referencia a su estructura interna
- El probador considera al objeto de prueba como una caja negra



IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Técnicas de caja negra. Características. Información adicional.

- Las condiciones de prueba, casos de prueba y datos de prueba se deducen de una base de prueba que puede incluir requisitos de software, especificaciones, casos de uso e historias de usuario.
- Los casos de prueba se pueden utilizar para detectar diferencias entre los requisitos y la implementación de los requisitos, así como desviaciones con respecto a los requisitos.
- La cobertura se mide en función de los elementos probados en la base de prueba y de la técnica aplicada a la base de prueba.

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Técnicas de Prueba de Caja Blanca

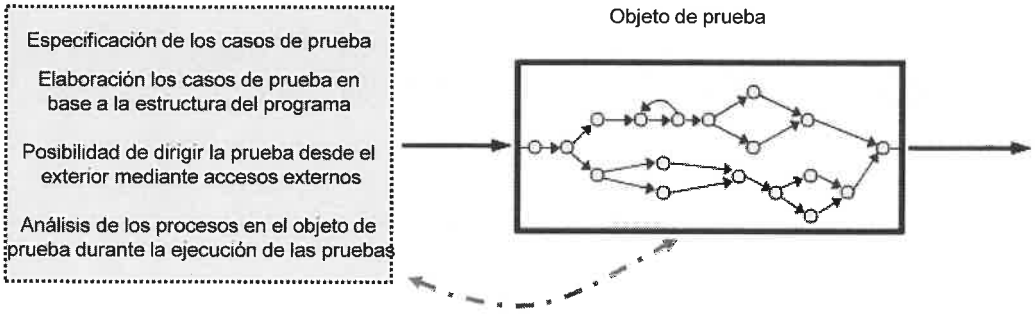
- Las técnicas de prueba de caja blanca (también conocidas como **técnicas basadas en la estructura**) se basan en un análisis de la estructura interna y el procesamiento del objeto de prueba.
- Como los casos de prueba dependen de cómo esté diseñado el software, sólo pueden crearse **después** del diseño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

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Técnicas de Prueba de Caja Blanca. Información adicional

- También llamadas técnicas estructurales o basadas en la estructura
- Se basan en un análisis de la arquitectura, el diseño detallado, la estructura interna o el código del objeto de prueba.
- A diferencia de las técnicas de prueba de caja negra, las técnicas de prueba de caja blanca se concentran en la estructura y el procesamiento dentro del objeto de prueba.



IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Técnicas de Prueba de Caja Blanca. Información adicional

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

- El probador conoce la estructura del programa y su código, esto es, jerarquía de componentes, flujo de control , flujo de datos etc..
- Los casos de prueba se especifican en base a la estructura del programa / código del programa
- Los accesos al objeto de pruebas durante la ejecución son posibles

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Técnicas de Prueba Basadas en la Experiencia

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

Técnicas de Prueba Basadas en la Experiencia. Información adicional

Características

- Las condiciones de prueba, los casos de prueba y los datos de prueba se deducen de una base de prueba que puede incluir el conocimiento y la experiencia de probadores, desarrolladores, usuarios y otros implicados
- Este conocimiento y experiencia incluye el uso esperado del software, su entorno, los posibles defectos y la distribución de dichos defectos.
- El estándar internacional (ISO/IEC/IEEE 29119-4) contiene descripciones de técnicas de prueba y sus correspondientes medidas de cobertura

IV – Análisis y Diseño de la Prueba
4.1 Introducción a las Técnicas de Prueba

Ejemplo OA FL-4.1.1.

- Las pruebas de sistema realizadas por equipos de prueba independientes sobre aspectos funcionales suelen ser un ejemplo habitual de aplicación de técnicas de caja negra, pero … ¡¡no deben confundirse !! (técnicas de caja negra, pruebas de sistema y pruebas funcionales son conceptos distintos)
- La utilización de herramientas de cobertura y de métricas de código por los programadores es uno de los ejemplos de uso más habitual de técnicas de caja blanca
- La realización de pruebas exploratorias por probadores en un desarrollo software en contexto ágil, es un ejemplo habitual de prueba basada en la experiencia.
- Durante el desarrollo del presente capítulo, se profundizará en los detalles y en ejemplos sobre cada uno de estos tipos de técnicas

IV – Análisis y Diseño de la Prueba

4.1 Introducción a las Técnicas de Prueba

Información adicional. Clasificación de técnicas de pruebas

Prueba dinámica
- Análisis del software mediante ejecución con datos de prueba

vs.

Prueba estática
- Análisis del software sin ejecución

Caja negra
- Implementación desconocida durante la especificación de casos de prueba
- La especificación es el punto de partida para la definición de casos de prueba

vs.

Caja blanca
- Implementación conocida durante la especificación de casos de prueba
- Generación de casos de prueba mediante el análisis de la implementación

IV – Análisis y Diseño de la Prueba

4.2 Técnicas de Prueba de Caja Negra

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

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

Otras técnicas de caja negra son, p.ej. (no tratadas en esta formación y fuera del alcance de la certificación):

- Casos de uso
- Pruebas aleatorias
- Análisis de gráficos de causa y efecto
- Pairwise Testing (pruebas de a pares)

4.2.1 Partición de Equivalencia

La Partición de Equivalencia (PE) divide los datos en particiones (conocidas como particiones de equivalencia) basándose en la expectativa de que todos los elementos de una partición determinada van a ser procesados de la misma manera por el objeto de prueba.

- La teoría que subyace a esta técnica es que, si un caso de prueba, que prueba un valor de una partición de equivalencia, detecta un defecto, este defecto también debería ser detectado por los casos de prueba que prueban cualquier otro valor de la misma partición.
- Por lo tanto, **basta con una prueba para cada partición**.

Nota: Es posible encontrar bibliografía, referencias y ejemplos donde se usa el término "**clase de equivalencia**" como sinónimo de "partición de equivalencia".

4.2.1 Partición de Equivalencia

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

4.2.1 Partición de Equivalencia

Partición de Equivalencia – procedimiento de actuación

- Las particiones de equivalencia se identifican para datos de entrada válidos y no válidos
  - Si se define un valor x como "x > 0", aparecen en principio dos particiones de equivalencia – una todas las entradas "> 0" y otra las entradas "<= 0"
  - Pueden identificarse tipos de datos no numéricos – esto representa otra posible partición de equivalencia
- Todos los tipos de entrada (elementos) del objeto de prueba son identificadas
  - Campos de la máscara de una GUI
  - Parámetros de una función
- Para cada una de las entradas se definen el rango de los valores donde el comportamiento es idéntico (p.e la fórmula aplicada es la misma)
  - Cada partición de equivalencia válida tendrá un comportamiento idéntico del objeto de prueba para todos los valores de la partición realizando el comportamiento definido
  - Los valores de cada partición de equivalencia inválida provocarán el mismo comportamiento del objeto de prueba (p.e mostrar un mensaje de entrada inválida)
- Se prueba con un solo representante de cada partición de equivalencia
  - Para todas las pruebas con otro valor de la misma partición de equivalencia (sea válida o inválida) se espera un comportamiento idéntico

IV – Análisis y Diseño de la Prueba

4.2.1 Partición de Equivalencia

Partición de Equivalencia – procedimiento de actuación

- **Ejemplo** OA FL-4.2.1**:**
- Un programa precisa la entrada de un peso en "Kg." (una función integrada redondea siempre sin decimales – valores menores que "0,5" se redondean a "0"). Se incluye el cero
- Se permiten aquí valores ≥ 0 y no permitidos serían todos los números negativos y los valores de entrada no numéricos (p.ej., "Hugo")

  - Partición de equivalencia válida: $x \geq 0$
  - 1. Partición de equivalencia no válida: $x < 0$
  - 2. Partición de equivalencia no válida : x = no numérico

Partición de Equivalencia – procedimiento de actuación

- **Ejemplo** OA FL-4.2.1 :
- Junto a las particiones de equivalencia no válidas enunciadas anteriormente existen más particiones de equivalencia no válidas que no se tienen en cuenta en este ejemplo:
  - x > 0, pero mayor que el mayor número que puede representar el ordenador (un número tan grande que supera la capacidad de la variable)
  - x > 0, pero más pequeño que el número positivo más pequeño que se puede representar en el ordenador

En general se podrían identificar dos tipos de particiones de equivalencia no válidas

1.- Del mismo tipo de dato al que pertenecen los valores de entrada correspondientes a la partición válida, pero se encuentran fuera de rango.

2.- De un tipo de dato distinto al que pertenecen los valores de entrada correspondientes a la partición válida. En este caso, el tipo dato inválido elegido es un representante de todos los tipos inválidos. No es que no se tengan en cuenta otros tipos inválidos, se utiliza una única partición para representar a todos los tipos inválidos donde el objeto de prueba tiene el mismo comportamiento

IV – Análisis y Diseño de la Prueba
4.2.1 Partición de Equivalencia

Partición de Equivalencia – procedimiento de actuación

- Durante la formación de particiones de equivalencia se consideran los rangos de definición de los valores
  - Valores de entrada y salida respectivamente del programa
- Los valores definidos se dividen en particiones de equivalencia
  - Esto es, aquellos para los que se espera un comportamiento idéntico del sistema son agrupados en una partición de equivalencia (CE)
- Las particiones de equivalencia de cada entrada (de cada elemento) deben ser subdivididas
  - CE válida: dentro del rango de definición se agrupan todos los valores que se procesan de forma idéntica por el objeto de prueba
    - En ejemplo V/03-1 son todas las entradas numéricas positivas
  - CE no válidas : fuera del rango de definición se distinguen dos casos:
    - Valores con formato correcto pero que están fuera del rango de definición se agrupan en una o más particiones de equivalencia
    - Valores con formato incorrecto forman generalmente otra partición de equivalencia
    - En el ejemplo V/03-1 se forman, p.ej., dos particiones de equivalencia no válidas: x < 0 y x no es un valor numérico

Partición de Equivalencia – procedimiento de actuación

- Finalmente se elige un representante para cada CE y se define el resultado esperado
  - Para el ejemplo V/03-1 :

Variable	Partición de equivalencia	Representante
Peso	CE_1: x ≥ 0	+1,00
	CE_2: x < 0	-1,00
	CE_3: x no numérico	Hugo

Partición de Equivalencia – procedimiento de actuación

- **Ejemplo** OA FL-4.2.1 **:**
  - Un programa calcula el precio de un producto en función de su valor, un descuento en % y gastos de envío establecidos de (6,- , 9,- o 12,-€ según envío)

Variable	Particiones de equivalencia	Estado	Representante
Valor del producto	CE_{11}: x >= 0	Válido	1000,00
	CE_{12}: x < 0	No válido	-1000,00
	CE_{13}: x no numérico	No válido	Hugo
Descuento	CE_{21}: 0% ≤ x ≤ 100%	Válido	10%
	CE_{22}: x < 0%	No válido	-1%
	CE_{23}: x > 100%	No válido	101%
	CE_{24}: x no numérico	No válido	Hugo
Gastos de envío	CE_{31}: x = 6	Válido	6
	CE_{32}: x = 9	Válido	9
	CE_{33}: x = 12	Válido	12
	CE_{34}: x ≠ {6, 9, 12}	No válido	5
	CE_{35}: x no numérico	No válido	Hugo

Existen las siguientes excepciones:
- Valor del producto numérico positivo con dos decimales
- Descuento numérico en formato de % sin comas, entre 0% y 100%
- Para los gastos de envío se proporcionan valores fijos que pueden ser 6, 9, o 12

Partición de Equivalencia – procedimiento de actuación

- **Ejemplo** OA FL-4.2.1 **:**
  - Las particiones de equivalencia válidas permiten formar las siguientes combinaciones o casos de prueba (CP01, CP02 y CP03) :

Variable	Calse de equivalencia	Estado	Representante	CP01	CP02	CP03
Valor del producto	CE_{11}: x >= 0	Válido	1000,00	●	●	●
	CE_{12}: x < 0	No válido	-1000,00			
	CE_{13}: x no numérico	No válido	Hugo			
Descuento	CE_{21}: 0% < x < 100%	Válido	10%	●	●	●
	CE_{22}: x < 0%	No válido	-1%			
	CE_{23}: x > 100%	No válido	101%			
	CE_{24}: x no numérico	No válido	Hugo			
Gastos de envío	CE_{31}: x = 6	Válido	6	●		
	CE_{32}: x = 9	Válido	9		●	
	CE_{33}: x = 12	Válido	12			●
	CE_{34}: x ≠ {6, 9, 12}	No válido	5			
	CE_{35}: x no numérico	No válido	Hugo			

IV – Análisis y Diseño de la Prueba
4.2.1 Partición de Equivalencia

Partición de Equivalencia – procedimiento de actuación

- **Ejemplo :**
  - De las CE no válidas se deducen directamente los siguientes casos de prueba: cada una en combinación con CE válidas de los otros elementos

Variable	Calse de equivalencia	Estado	Represen-tante	CP04	CP05	CP06	CP07	CP08	CP09	CP10
Valor del producto	CE_{11}: x >= 0	Válido	1000,00			●	●	●	●	●
	CE_{12}: x < 0	No válido	-1000,00	●						
	CE_{13}: x no numérico	No válido	Hugo		●					
Descuento	CE_{21}: 0% < x < 100%	Válido	10%	●	●				●	●
	CE_{22}: x < 0%	No válido	-1%			●				
	CE_{23}: x > 100%	No válido	101%				●			
	CE_{24}: x no numérico	No válido	Hugo					●		
Gastos de envío	CE_{31}: x = 6	Válido	6	●	●	●	●	●		
	CE_{32}: x = 9	Válido	9							
	CE_{33}: x = 12	Válido	12							
	CE_{34}: x ≠ {6, 9, 12}	No válido	5						●	
	CE_{35}: x no numérico	No válido	Hugo							●

Partición de Equivalencia – procedimiento de actuación

- **Ejemplo :**
  - Se deducen en total 10 casos de prueba
    3 casos válidos y 7 casos no válidos:

Variable	Particiones de equivalencia	Estado	CP01	CP02	CP03	CP04	CP05	CP06	CP07	CP08	CP09	CP10
Valor del producto	Válido	1000,00	•	•	•			•	•	•	•	•
	No válido	-1000,00				•						
	No válido	Hugo					•					
Descuento	Válido	10%	•	•	•	•	•				•	•
	No válido	-1%						•				
	No válido	101%							•			
	No válido	Hugo								•		
Gastos de envío	Válido	6	•			•	•	•	•	•		
	Válido	9		•								
	Válido	12			•							
	No válido	5									•	
	No válido	Hugo										•

Partición de Equivalencia – procedimiento de actuación

Información adicional

- Formación de particiones de equivalencia en base a los valores de salida
  - El procedimiento se puede aplicar de forma análoga para la división de todos los valores de salida definidos
  - La variable (el elemento) será entonces un valor de salida (p.ej., un campo de salida de una GUI)
  - De los posibles valores de salida definidos se formarán otra vez las respectivas particiones de equivalencia
    - Una CE engloba a aquellos valores a los que se subordina un mismo comportamiento de salida
  - Por cada partición de equivalencia se elige un representante
  - Se decide finalmente mediante qué valores de entrada se pueden conseguir los representantes
- Podría requerir un esfuerzo adicional significativo tener que buscar datos de entrada para los diferentes comportamientos de salida definidos

Partición de Equivalencia – Generalidades

- Descomposición
  - La mayor o menor utilidad de los casos de prueba generados depende de la precisión con que se descompongan las entradas y salidas individuales
  - Las PE no identificadas (olvidadas) durante el proceso implican el riesgo de no identificar posibles defectos

IV – Análisis y Diseño de la Prueba
4.2.1 Partición de Equivalencia
Partición de Equivalencia – Generalidades

- Elección de valores representantes de las CE
  - Cada valor representante de una CE puede ser elegido para el caso de prueba. Podrían usarse diversos criterios como: valores más frecuentes, valores que resultaron más problemáticos en el pasado, valores más interesantes para el negocio, etc.
  - Con el fin de evitar el enmascaramiento de defectos no se deben elegir valores representantes de CE no válidas junto a representantes de otras CE no válidas dentro del mismo caso de prueba. Usar un único valor representante de CE no válida por caso de prueba podría ayudar a evitarlo

Partición de Equivalencia – Generalidades

- El criterio de salida de las pruebas podría ser un determinado grado de cobertura de las CE
  - ¿Cuántas de las posibles CE han sido probadas en relación a todas las CE definidas?

$$\text{Cobertura de CE} = \frac{\text{Nº de CE probadas}}{\text{Nº total de CE}} * 100\ \%$$

IV – Análisis y Diseño de la Prueba
4.2.1 Partición de Equivalencia

Partición de Equivalencia – Generalidades

- Paso de definiciones / requisitos a clases de equivalencia
  - A menudo una tarea difícil ya que los documentos disponibles no son precisos o son incompletos
  - Límites de rangos de valores imprecisos o, eventualmente falta de datos acerca de la definición exacta dificultan la transformación
  - ¡A menudo suele ser necesario volver a consultar con el cliente!

4.2.1 Partición de Equivalencia

Partición de Equivalencia – Generalidades

- Ventajas del procedimiento
  - Determinación sistemática de casos de prueba, es decir, con un número mínimo de casos de prueba se maximiza la cobertura de pruebas
  - La misma técnica permite establecer el número de casos de prueba necesarios para lograr distintos porcentajes de cobertura.
  - Esta técnica permite asignar prioridades a particiones de equivalencia y, por lo tanto, es posible utilizar estas prioridades a efectos de organizar la ejecución de casos, juegos y procedimientos de prueba.
  - Esta técnica es una técnica débil y, además, permite evaluar el tratamiento de excepciones y errores. Lo que no se podría afirmar es en qué medida cubre o puede cubrir estos tratamientos.

4.2.2 Análisis del Valor Frontera

El **Análisis del Valor Frontera (AVF)** es una técnica basada en practicar las fronteras de las particiones de equivalencia.

- Por lo tanto, el AVF sólo puede utilizarse para particiones ordenadas.
- **Los valores mínimo y máximo de una partición son sus valores frontera**.
- En el caso de AVF, si dos elementos pertenecen a la misma partición, todos los elementos entre ellos deben pertenecer también a esa partición.
- El AVF se concentra en los valores frontera de las particiones porque es más probable que los desarrolladores cometan errores con estos valores frontera.
- Los **defectos típicos** que encuentra el AVF se localizan cuando las fronteras implementadas se colocan erróneamente en posiciones superiores o inferiores a las previstas o se omiten por completo.

4.2.2 Análisis del Valor Frontera

Esta formación abarca dos versiones del AVF:

- AVF de 2 valores y
- AVF de 3 valores.

Se diferencian en cuanto a los elementos de cobertura por frontera que necesitan ser practicados para lograr una cobertura del 100%.

IV – Análisis y Diseño de la Prueba

4.2.2 Análisis del Valor Frontera

En el **AVF de 2 valores** (Craig 2002, Myers 2011), para cada valor frontera hay dos elementos de cobertura: **este valor frontera** y **su vecino más cercano** perteneciente a la partición adyacente.

- Para lograr una cobertura del 100% con AVF de 2 valores, los casos de prueba deben utilizar todos los elementos de cobertura, es decir, todos los valores frontera identificados.
- La cobertura se mide como el número de valores frontera que se han practicado, dividido por el número total de valores frontera identificados, y se expresa en porcentaje.

4.2.2 Análisis del Valor Frontera

En el AVF de 3 valores (Koomen 2006, O'Regan 2019), para cada valor frontera hay tres elementos de cobertura: este **valor frontera** y **sus dos vecinos**.

- Por lo tanto, en el AVF de 3 valores algunos de los elementos de cobertura pueden no ser valores frontera.
- Para lograr una cobertura del 100% con AFV de 3 valores, los casos de prueba deben practicar todos los elementos de cobertura, es decir, los valores frontera identificados y sus vecinos.
- La cobertura se mide como el número de valores frontera y sus vecinos practicados, dividido por el número total de valores frontera identificados y sus vecinos, y se expresa en porcentaje.
- El **AVF de 3 valores es más riguroso que el AVF de 2 valores**, ya que puede detectar defectos pasados por alto por el AVF de 2 valores.
  - Por ejemplo (OA FL-4.2.2), si la decisión "si (x = 10) ..." se implementa incorrectamente como "si (x = 10) ...", ningún dato de prueba derivado del AVF de 2 valores (x = 10, x = 11) puede detectar el defecto. Sin embargo, es probable que x = 9, derivado del AVF de 3 valores, lo detecte

IV – Análisis y Diseño de la Prueba

4.2.2 Análisis del Valor Frontera

Ejemplo OA FL-4.2.2:

Vamos a analizar el siguiente **escenario**: Suponer que un campo de entrada acepta un único valor entero como entrada, utilizando un teclado numérico para limitar las entradas de modo que las entradas no enteras sean imposibles. El rango válido es de 1 a 5, ambos valores incluidos. El resultado sería:

- Hay tres particiones de equivalencia: inválida (demasiado baja); válida; inválida (demasiado alta).
- Para la partición de equivalencia válida, los valores frontera son 1 y 5
- Para la partición no válida (demasiado alta), los valores frontera son 6 y 9.
- Para la partición inválida (demasiado baja), sólo hay un valor frontera, 0, porque se trata de una partición con un solo miembro.
- En el ejemplo anterior, se identifican dos valores frontera por frontera
- La frontera entre inválido (demasiado bajo) y válido proporciona los valores de prueba 0 y 1.
- La frontera entre válido e inválido (demasiado alto) proporciona los valores de prueba 5 y 6.

IV – Análisis y Diseño de la Prueba

4.2.2 Análisis del Valor Frontera

- En el ejemplo anterior, utilizando tres puntos para los valores frontera, los valores de la prueba de la frontera inferior son 0, 1 y 2, y los valores de la prueba de la frontera superior son 4, 5 y 6.

¡La experiencia muestra que los defectos son más frecuentes en las fronteras!

- Si las clases de equivalencia de una variable vienen representadas en forma de un grupo de valores no se podrá, en general, formar valores límite
  - P.ej. Soltero, casado, separado, viudo

IV – Análisis y Diseño de la Prueba

4.2.2 Análisis de Valores Frontera

- **Ejemplo** OA FL-4.2.2 **:**
  - Rango de valores para el descuento en %: $0{,}00 \leq x \leq 100{,}00$
  - **Formación de CE**
    3 Clases (< 0, Rango de definición, > 100)
    p.ej., con los representantes: *-50,00*; *50,00*; *150,00*
  - **Análisis de valores límite**
    amplía los representantes a :
    1. (*-50,00*): *-0,01*
    2. (*50,00*): *0,00*; *100,00*
    3. (*150,00*): *100,01*

4.2.3 Prueba de Tabla de Decisión

Las **tablas de decisión** se utilizan para probar la implementación de los requisitos de sistemas que especifican cómo diferentes **combinaciones** de condiciones dan lugar a diferentes resultados.

- Las tablas de decisión son una forma efectiva de registrar lógicas complejas, como las reglas de negocio.

Una tabla de decisión es:

- Una tabla que muestra lo que el programa debería hacer bajo combinaciones de eventos relevantes.
- Una tabla que muestra la lógica del programa
- Parecido a un árbol de decisión en la forma en que lista la información.
  - Nota: El concepto de árbol de decisión no pertenece al programa de estudios de ISTQB Nivel Básico y se incluye como conocimiento adicional. El concepto se utiliza en el programa de estudios de ISTQB Analista de prueba donde se amplía el presente apartado.

IV – Análisis y Diseño de la Prueba

4.2.3 Prueba de Tabla de Decisión

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

4.2.3 Prueba de Tabla de Decisión

La **notación** para las condiciones es la siguiente:

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

Aunque la notación anterior es la más habitual, también se pueden utilizar otras notaciones. Algunos ejemplos podrían ser:

- El género de una persona
- Colores
- Clasificación o cuantificación: Alto, Medio, Bajo

IV – Análisis y Diseño de la Prueba

4.2.3 Prueba de Tabla de Decisión

- Una **tabla de decisión completa** tiene suficientes columnas para cubrir todas las combinaciones de condiciones.
- La tabla puede **simplificarse** eliminando las columnas que contengan combinaciones de condiciones inviables.
- La tabla también se puede minimizar fusionando en una sola columna las columnas en las que algunas condiciones no afectan al resultado.
- Estas tablas simplificadas y/o reducidas de denominan **colapsadas**
- Los algoritmos de minimización de tablas de decisión están fuera del alcance de esta formación y del examen de certificación.

IV – Análisis y Diseño de la Prueba

4.2.3 Prueba de Tabla de Decisión

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

IV – Análisis y Diseño de la Prueba

4.2.3 Prueba de Tabla de Decisión

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

Prueba de Tabla de Decisión. Ejemplo OA FL-4.2.3

- Es un buen sistema para representar la lógica de los requisitos de usuario, sobre todo las reglas de negocio complejas.
- Las tablas de decisión están compuestas por cuatro secciones:

	REGLAS DE DECISIÓN
IDENTIFICACIÓN DE CONDICIONES	VALORES DE CONDICIONES
IDENTIFICACIÓN DE ACCIONES	VALORES DE ACCIONES

- Una vez confeccionada la tabla, quedarán determinadas las reglas de decisión. Estas son proposiciones que se leerán verticalmente, partiendo desde la sección Valores de Condiciones y descendiendo por la sección Valores de Acciones. Se las enuncia así: "SI.....(*condición1, condición2, etc.*)... ENTONCES ... (*acción1, acción2, etc.*)....".

Prueba de Tabla de Decisión. Ejercicio OA FL-4.2.3

	REGLAS DE DECISIÓN			
CONDICIONES	1	2	3	4
¿Pago contado?	S	S	N	N
¿Compra > 50.000€?	S	N	S	N
ACCIONES				
Calcular descuento del 5% sobre el importe de la compra	X	X		
Calcular descuento del 7% sobre el importe de la compra	X		X	
Calcular importe neto de la factura	X	X	X	X

¿Cuál sería el comportamiento del sistema para un pago de 50.000€ con un pagaré? (es decir, un pago que no es al contado)

4.2.4 Prueba de Transición de Estado

Un **diagrama de transición de estados** modela el comportamiento de un sistema mostrando sus posibles **estados** y las **transiciones de estado válidas**.

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

4.2.4 Prueba de Transición de Estado

Una **tabla de estados** es un modelo equivalente a un diagrama de transición de estado.

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

4.2.4 Prueba de Transición de Estado

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

- Un caso de prueba puede cubrir, y normalmente cubrirá, varias transiciones entre estados.
- Existen muchos criterios de cobertura para las pruebas de transición de estado.
- Esta formación analiza tres de ellos:
  - cobertura de todos los estados
  - cobertura de transiciones válidas
  - cobertura de todas las transiciones

IV – Análisis y Diseño de la Prueba

4.2.4 Prueba de Transición de Estado

En la **cobertura de todos los estados**, los elementos de cobertura son los estados.

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

4.2.4 Prueba de Transición de Estado

En la **cobertura de transiciones válidas** (también denominada **cobertura de conmutador 0**), los elementos de cobertura son transiciones válidas únicas.

- Para lograr una cobertura de transiciones válidas del 100%, los casos de prueba deben practicar todas las transiciones válidas.
- La cobertura se mide como el número de transiciones válidas practicadas dividido por el número total de transiciones válidas, y se expresa en porcentaje.

4.2.4 Prueba de Transición de Estado

En la **cobertura de todas las transiciones**, los elementos de cobertura son todas las transiciones que aparecen en una tabla de estados.

- Para lograr una cobertura del 100% de todas las transiciones, los casos de prueba deben practicar todas las transiciones válidas e intentar ejecutar las transiciones no válidas.

4.2.4 Prueba de Transición de Estado

Probar sólo una transición no válida en un único caso de prueba ayuda a evitar el **enmascaramiento de defecto**, es decir, *una situación en la que un defecto impide la detección de otro.*

- La cobertura se mide como el número de transiciones válidas e inválidas practicadas o intentadas por los casos de prueba ejecutados, dividido por el número total de transiciones válidas e inválidas, y se expresa en porcentaje.

IV – Análisis y Diseño de la Prueba

4.2.4 Prueba de Transición de Estado

- La **cobertura de todos los estados** es más débil que la cobertura de las transiciones válidas, porque normalmente puede lograrse sin practicar todas las transiciones.
- La **cobertura de transiciones válidas** es el criterio de cobertura más utilizado.

- *Lograr una cobertura completa de transiciones válidas garantiza una cobertura completa de todos los estados.*
- *Lograr la cobertura completa de todas las transiciones garantiza tanto la cobertura completa de todos los estados como la cobertura completa de las transiciones válidas y debería ser un requisito mínimo para el software de misión y seguridad física críticas.*

IV – Análisis y Diseño de la Prueba

4.2.4 Prueba de Transición de Estado

Prueba de Transición de Estado. Ejemplo OA FL-4.2.4

Para aquellos sistemas que se ajusten a un modelo de transición de estados, el objetivo de las pruebas sobre este tipo de sistemas sería demostrar la conformidad de los requisitos especificados en forma de diagrama de transición de estados o tablas de transición de estados.

ESTADOS	EVENTOS		
	Evento_1		Evento_n
Estado_1	Estado_2 Acción_1 (1)		Ignorar (n)
Estado_m	Ignorar (n*(m-1)+1)		Ignorar (n*m)

Se diseñarán casos de prueba que cubran todas las transiciones de estados representadas en la tabla.

IV – Análisis y Diseño de la Prueba

4.2.4 Prueba de Transición de Estado

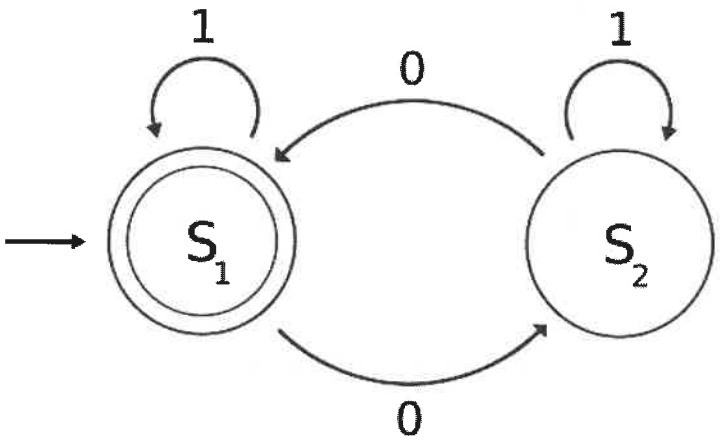
Prueba de Transición de Estado. Ejercicio OA FL-4.2.4

Se muestra una tabla de transición de estados para una máquina junto con el correspondiente diagrama de estados. ¿puedes explicar el comportamiento?

Tabla de Transición de Estados

Entrada / Estado	1	0
S_1	S_1	S_2
S_2	S_2	S_1

Diagrama de estados



Todas las entradas posibles a la máquina están enumeradas a través de las columnas de la tabla. Todos los estados posibles están enumerados a través de las filas. Desde la tabla de transición de estados anterior, es fácil ver que si la máquina está en **S1** (la primera fila), y la siguiente entrada es el carácter **1**, la máquina permanecerá en **S1**. Si llega un carácter **0**, la máquina realizará la transición a **S2** como puede verse desde la segunda columna. En el diagrama esto es denotado por la flecha desde **S1** a **S2** etiquetada con un **0**.

4.2.4 Prueba de Transición de Estado

Prueba de Transición de Estado. Información adicional

- Ejemplo de criterio mínimo de finalización de pruebas:
  - Cada estado se debe alcanzar al menos una vez
  - Cada transición de estados se debe alcanzar al menos una vez
- Ventajas e inconvenientes del procedimiento
  - Facilita probar objetos de prueba dependientes del estado. Por ejemplo, podría aplicarse a un objeto de prueba con un ciclo de vida basado en estados (nuevo, asignado, corregido, comprobado, cerrado)
  - La determinación de los estados suele ser la parte más laboriosa. Por ejemplo, los estados son en ocasiones muy complejos en su estructura, esto es, son determinados por un gran número de parámetros diferentes.
  - Tanto la definición de casos de prueba como la valoración de las pruebas son muy laboriosas en estos casos
  - La cobertura de todas las transiciones de estado no garantiza unas pruebas completas

4.3 Técnicas de Prueba de Caja Blanca

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

- Prueba de sentencia
- Pruebas de rama

Existen técnicas más rigurosas que se utilizan en algunos entornos de seguridad crítica, misión crítica o alta integridad para lograr una cobertura de código más profunda.

También hay técnicas de prueba de caja blanca que se utilizan en niveles de prueba superiores (por ejemplo, prueba de API), o que utilizan una cobertura no relacionada con el código (por ejemplo, la cobertura de neuronas en la prueba de redes neuronales).

Estas técnicas mencionadas en los dos párrafos anteriores no se tratan en esta formación y por tanto están fuera del alcance del examen de certificación.

Información adicional

- Las técnicas de prueba de caja blanca se basan en la estructura interna del objeto de prueba.
- Las técnicas de prueba de caja blanca se pueden utilizar en todos los niveles de prueba, pero las dos técnicas relacionadas con el código que se discuten en esta sección se utilizan con mayor frecuencia en el **nivel de prueba de componente**.
- Otras técnicas de caja blanca (fuera de alcance de la certificación) son p.ej., :
  - Cobertura de condiciones
  - Cobertura de rutas o caminos
  - LCSAJ (Linear Code Sequence And Jump)
  - Procedimiento basado en el flujo de datos

Técnicas de Prueba de Caja Blanca. Información adicional

- En las técnicas de caja blanca se trata de ejecutar partes del programa
  - En teoría deberían ser probadas todas las partes del programa al menos una vez durante las pruebas
  - La ejecución es realizada mediante **herramientas** (p.ej., mediante analizadores de cobertura):
- Se realiza una **instrumentación**, esto es, se instalan contadores en determinadas posiciones del objeto de prueba
  - Estos contadores están al principio de la prueba a cero y son aumentados escalonadamente con cada ejecución de la parte del programa
  - Si al final de la prueba quedan contadores a cero significa que la parte del programa no se ha ejecutado

Técnicas de Prueba de Caja Blanca. Información adicional

- ¡Siempre se prueba sólo lo que existe!
- Casos de prueba basados en la implementación disponible
- ¡La falta de funciones no se descubre!
- Utilización de las técnicas más potentes en las fases de desarrollo mas tempranas
- Pruebas cercanas al desarrollo con las técnicas de caja blanca
- P.ej., pruebas de componentes y de integración
- Los procedimientos se diferencian en la intensidad de pruebas
- En función del procedimiento y la estructura del programa varía el número de casos de prueba
- **En todos los niveles de prueba**, pero especialmente en las pruebas de componentes y en las de integración de componentes, se pueden utilizar herramientas para medir la cobertura de código de elementos como sentencias o decisiones.

En la **prueba de sentencia**, los elementos de cobertura son sentencias ejecutables.

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

Ejemplo OA FL-4.3.1 :

- Veamos a continuación los resultados de utilizar una herramienta de apoyo a esta técnica. Dinámica dirigida por el profesor.

4.3.1 Prueba de Sentencia y Cobertura de Sentencia

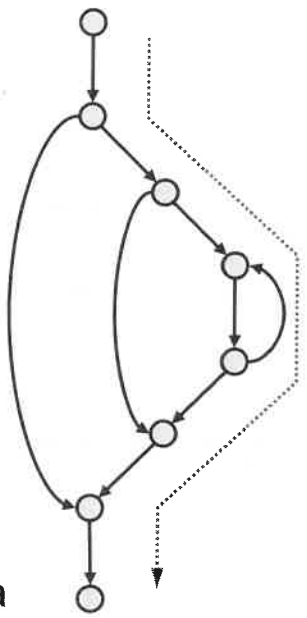
- Cuando se alcanza el 100% de cobertura de sentencias, se asegura que todas las sentencias ejecutables del código han sido practicadas al menos una vez.
- En concreto, esto significa que se ejecutará cada sentencia con un defecto, lo que puede provocar un fallo que demuestre la presencia del defecto.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

4.3.1 Prueba de Sentencia y Cobertura de Sentencia

Ejemplo de cobertura de sentencia. No aplica al examen

Segmento de programa representado a la derecha como gráfico de flujo de control :

```
if (i > 0) {
    j = f(i);
    if (j > 10) {
        for (k=i; k > 10; k --) {
            ...
        }
    }
}
```



Vemos que es posible recorrer todas las sentencias cuando la ejecución se realiza por la línea punteada.

IV – Análisis y Diseño de la Prueba

4.3.2 Prueba de Rama y Cobertura de Rama

- Una rama es una transferencia de control entre dos nodos del grafo del flujo de control, que muestra las posibles secuencias en las que se ejecutan las sentencias del código fuente en el objeto de prueba.
- Cada transferencia de control puede ser incondicional (es decir, código en línea recta) o condicional (es decir, un resultado de la decisión).
- En la prueba de rama, los elementos de cobertura son las ramas y el objetivo es diseñar casos de prueba que permitan practicar las ramas del código hasta alcanzar un nivel de cobertura aceptable.
- La cobertura se mide como el número de ramas practicadas por los casos de prueba dividido por el número total de ramas, y se expresa en porcentaje.

IV – Análisis y Diseño de la Prueba

4.3.2 Prueba de Rama y Cobertura de Rama

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

4.3.2 Prueba de Rama y Cobertura de Rama

Ejemplo OA FL-4.3.2.

- Veamos a continuación los resultados de utilizar una herramienta de apoyo a esta técnica. Dinámica dirigida por el profesor.

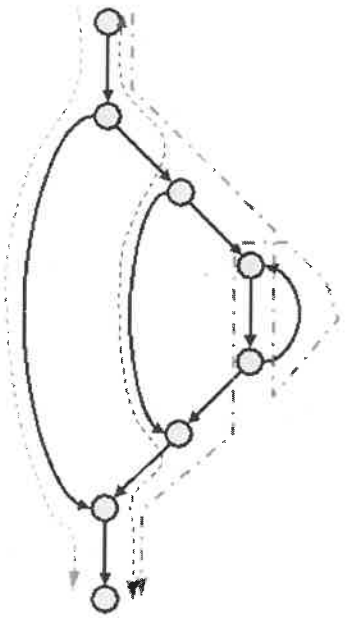
IV – Análisis y Diseño de la Prueba

4.3.2 Prueba de Rama y Cobertura de Rama

Ejemplo de cobertura de rama. No aplicable al examen

Segmento de programa representado a la derecha como gráfico de flujo de control :

```
if (i > 0) {
    j = f(i);
    if (j > 10) {
        for (k=i; k > 10; k --) {
            ...
        }
    }
}
```



Nota: La cobertura de rama de denominaba **cobertura de decisión** en anteriores versiones del programa de estudios ISTQB nivel básico

4.3.3 El valor de la Prueba de Caja Blanca

- Un **punto fuerte** fundamental que comparten todas las técnicas de caja blanca es que durante las pruebas se tiene en cuenta toda la implementación del software, lo que facilita la detección de defectos incluso cuando la especificación del software es vaga, obsoleta o incompleta.
- Un **punto débil** correspondiente es que si el software no implementa uno o más requisitos, la prueba de caja blanca puede **no detectar los defectos de omisión** resultantes (Watson 1996).
- Las técnicas de caja blanca pueden utilizarse en la prueba estática (por ejemplo, durante la realización de simulacros de código).
- Son muy adecuadas para revisar código que aún no está listo para su ejecución (Hetzel 1988), así como pseudocódigo y otra lógica de alto nivel o descendente que pueda modelarse con un grafo del flujo de control.

IV – Análisis y Diseño de la Prueba

4.3.3 El valor de la Prueba de Caja Blanca

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

Ejemplo OA FL-4.3.3.

- Como en ocasiones anteriores, es más claro poner un contraejemplo para explicar la prueba de caja blanca. Se ha podido comprobar en proyectos reales que no realizar pruebas de caja blanca adecuadas está relacionado con serios problemas de mantenibilidad del software.
- Sería como no realizar nunca una revisión a un vehículo porque no detectas una avería.

IV – Análisis y Diseño de la Prueba

4.4 Técnicas de Prueba Basadas en la Experiencia

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

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

IV – Análisis y Diseño de la Prueba

4.4 Técnicas de Prueba Basadas en la Experiencia

Información adicional

- Al aplicar técnicas de prueba basadas en la experiencia, los casos de prueba se obtienen a partir de la competencia e **intuición** del probador y de su experiencia con aplicaciones y tecnologías similares.
- Estas técnicas pueden ser útiles para identificar pruebas que no fueron fácilmente identificadas por otras técnicas más sistemáticas.
- Dependiendo del enfoque y la **experiencia del probador**, estas técnicas pueden lograr grados muy diferentes de cobertura y efectividad.
- La **cobertura puede ser difícil de evaluar** y puede no ser medible con estas técnicas

4.4.1 Predicción de Errores

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

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

IV – Análisis y Diseño de la Prueba

4.4.1 Predicción de Errores

En general, los errores, defectos y fallos pueden estar relacionados con:

- la entrada (por ejemplo, no se acepta la entrada correcta, parámetros erróneos o ausentes),
- la salida (por ejemplo, formato incorrecto, resultado erróneo),
- la lógica (por ejemplo, casos ausentes, operador erróneo),
- el cálculo (por ejemplo, operando incorrecto, cálculo erróneo),
- las interfaces (por ejemplo, desajuste de parámetros,
- tipos incompatibles) o
- los datos (por ejemplo, inicialización incorrecta, tipo erróneo).

4.4.1 Predicción de Errores

Los **ataques de defecto** son un **enfoque metódico** de la implementación de la predicción de errores.

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

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

4.4.2 Prueba Exploratoria

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

IV – Análisis y Diseño de la Prueba

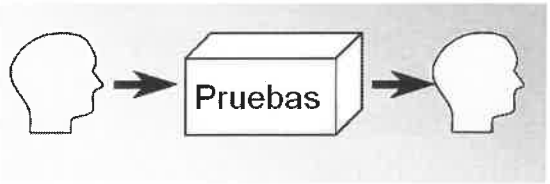
4.4 Técnicas de Prueba Basadas en la Experiencia

Propuesta de debate. Información adicional.

En pruebas sistemáticas (p.e.. usando técnicas de prueba de caja negra), las pruebas son diseñadas y registradas. Luego pueden ser ejecutadas posteriormente, incluso por otro probador

Basadas en términos:

- Cuantitativos, y
- Decisión

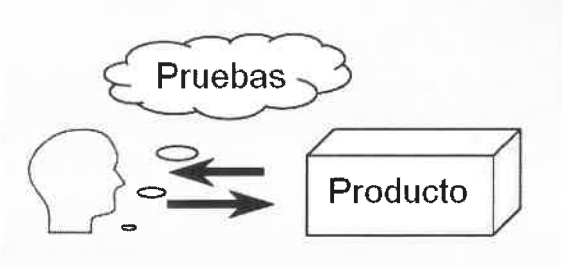


Vs

En las pruebas exploratorias, las pruebas son diseñadas y ejecutadas al mismo tiempo, y a menudo no son registradas

Basadas en términos de:

- Adaptación y
- Aprendizaje



IV – Análisis y Diseño de la Prueba

4.4 Técnicas de Prueba Basadas en la Experiencia

Ejemplo OA FL-4.4.1, OA FL-4.4.2, OA FL-4.4.3.

Es muy habitual que los participantes en los grupos de formación que tienen algunos años de experiencia en pruebas reconozcan que han utilizado de una u otra forma las técnicas basadas en la experiencia que se han expuesto. Aunque en algunos casos desconocían su nombre

- Cuando el equipo de pruebas “sabe” que una prueba es importante pero no tiene facilidad para explicar por qué podemos estar en una aplicación de predicción del error (algo nos dice que …)
- En los desarrollos ágiles es muy habitual realizar sesiones de investigación sobre el software para preparar un sprint. Podría ser un buen ejemplo de prueba exploratoria
- Todos hemos realizados comprobaciones habituales y sistemáticas de determinados aspectos del producto antes de iniciar las pruebas. Podría ser un buen ejemplo de prueba basada en lista de comprobación, aunque que se recomienda que sea lista tangible y no solo esté en nuestra cabeza.

4.5 Enfoques de Prueba Basados en la Colaboración

Cada una de las técnicas mencionadas anteriormente (véanse las secciones 4.2, 4.3 y 4.4) tiene un objetivo particular con respecto a la detección de defectos.

Los **enfoques basados en la colaboración**, en cambio, se concentran también en evitar defectos mediante la colaboración y la comunicación.

4.5.1 Redacción Colaborativa de Historias de Usuario

Una historia de usuario representa una prestación que será de valor para un usuario o comprador de un sistema o software.

Las historias de usuario tienen tres aspectos críticos (Jeffries 2000), denominados en conjunto las **"3 C"** (en referencia a las iniciales de los términos en inglés: Card, Conversation, Confirmation).

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

4.5.1 Redacción Colaborativa de Historias de Usuario

El formato más habitual de una historia de usuario es

- Como [rol],
- quiero que se cumpla [objetivo],
- de modo que pueda [valor de negocio resultante para el rol],
- seguido de los criterios de aceptación.

La autoría **colaborativa** de la historia de usuario puede utilizar técnicas como la tormenta de ideas y los mapas mentales.

La colaboración permite al equipo obtener una visión compartida de lo que debe entregarse, teniendo en cuenta tres perspectivas: el negocio, el desarrollo y la prueba.

IV – Análisis y Diseño de la Prueba

4.5.1 Redacción Colaborativa de Historias de Usuario

- Las buenas historias de usuario deben ser:
  - Independientes,
  - Negociables,
  - Valiosas,
  - Estimables,
  - Pequeñas y
  - Comprobables
- INVEST, en referencia a las iniciales de los términos en inglés: Independent, Negotiable, Valuable, Estimable, Small and Testable.
- Si un implicado no sabe cómo probar una historia de usuario, esto puede indicar que la historia de usuario no es lo suficientemente clara, o que no refleja algo valioso para él, o que simplemente necesita ayuda para probarla (Wake 2003).

.

Español		Inglés
Independiente	I	Independent
Negociable	N	Negotiable
Valioso/a	V	Valuable
Estimable	E	Estimable
Pequeño/a	S	Small
Capacidad de Prueba	T	Testable

IV – Análisis y Diseño de la Prueba
4.5.1 Redacción Colaborativa de Historias de Usuario

Ejemplo OA FL-4.5.1.

- La experiencia ha demostrado que la creación de una buena especificación (requisitos, historias de usuario) es similar a aprender Scrum o jugar al tenis: se explica en 30 minutos y se tardan años en dominarlo
- Existen excelentes estándares de referencia (https://www.ireb.org/), libros y publicaciones que describen la forma de especificar el producto.
- Pero el mecanismo que ha demostrado mayor tasa de éxito ha sido la realización de un taller (whorkshop), es decir, aprender haciendo.
- El profesor expondrá al grupo una breve descripción de la elaboración colaborativa de una historia de usuario ya que debe ser una experiencia que se viva y no un texto que se lea.

4.5.2 Criterios de Aceptación

Los **criterios de aceptación** de una historia de usuario son las condiciones que debe cumplir una implementación de la historia de usuario para ser aceptada por los implicados.

- Desde esta perspectiva, los criterios de aceptación pueden verse como las condiciones de prueba que deben ser practicadas por las pruebas.
- Los criterios de aceptación suelen ser un resultado de la Conversación (véase la sección 4.5.1).
- Los criterios de aceptación se utilizan para:
  - Definir el alcance de la historia de usuario.
  - Alcanzar un consenso entre los implicados.
  - Describir escenarios positivos y negativos.
  - Servir de base para la prueba de aceptación de usuario (véase la sección 4.5.3).
  - Permitir una planificación y estimación precisas.

IV – Análisis y Diseño de la Prueba

4.5.2 Criterios de Aceptación

Hay varias formas de redactar los criterios de aceptación de una historia de usuario.

Los dos formatos más comunes son:

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

La mayoría de los criterios de aceptación pueden documentarse en uno de estos dos formatos.

Sin embargo, el equipo puede utilizar otro formato personalizado, siempre que los criterios de aceptación estén bien definidos y sean inequívocos.

4.5.2 Criterios de Aceptación

Ejemplo OA FL-4.5.2.

En el programa de estudios de ISTQB Agile Tester se pueden encontrar un gran número de ejemplos de criterios de aceptación. Ejemplos:

- Realizar un análisis estático al 100% del código en prueba unitaria
- Se han reportado todos los defectos encontrados
- Se han cubierto todos los riesgos de calidad de conformidad con el alcance acordado de las pruebas
- No hay defectos importantes sin resolver (priorizados por riesgo e importancia)
- Las historias de usuario seleccionadas para la iteración están completas, el equipo las ha entendido y cuentan con criterios de aceptación detallados y testeables

4.5.3 Desarrollo Guiado por Prueba de Aceptación (DGPA)

DGPA es un enfoque de tipo **probar primero** (véase la sección 2.1.3).

- Los casos de prueba se crean antes de la implementación de la historia de usuario.
- Los casos de prueba son creados por miembros del equipo con diferentes perspectivas, por ejemplo, clientes, desarrolladores y probadores (Adzic 2009).
- Los casos de prueba pueden ejecutarse de forma manual o automatizada.

4.5.3 Desarrollo Guiado por Prueba de Aceptación (DGPA)

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

IV – Análisis y Diseño de la Prueba

4.5.3 Desarrollo Guiado por Prueba de Aceptación (DGPA)

- Dado que **ejemplos y pruebas son lo mismo**, estos términos suelen utilizarse indistintamente.
- Durante el diseño de prueba pueden aplicarse las técnicas de prueba descritas en las secciones 4.2, 4.3 y 4.4.
- Normalmente, los primeros casos de prueba son positivos, confirman el comportamiento correcto sin excepciones ni condiciones de error y comprenden la secuencia de actividades que se ejecutan si todo va como se espera.
- Una vez realizados los casos de prueba positivos, el equipo debe realizar pruebas negativas.
- Por último, el equipo debe cubrir también las características de calidad no funcionales (por ejemplo, la eficiencia de desempeño, la usabilidad).
- Los casos de prueba deben expresarse de forma comprensible para los implicados.

4.5.3 Desarrollo Guiado por Prueba de Aceptación (DGPA)

- Normalmente, los casos de prueba contienen frases en lenguaje natural que incluyen las precondiciones necesarias (si las hay), las entradas y las postcondiciones.
- Los casos de prueba deben cubrir todas las características de la historia de usuario y no deben ir más allá de la historia.
- Sin embargo, los criterios de aceptación pueden detallar algunos de los problemas descritos en la historia de usuario.
- Además, no debe haber dos casos de prueba que describan las mismas características de la historia de usuario.
- Cuando se capturan en un formato compatible con un marco de automatización de la prueba, los desarrolladores pueden automatizar los casos de prueba escribiendo el código de apoyo a medida que implementan la prestación descrita por una historia de usuario.
- Las pruebas de aceptación se convierten entonces en requisitos ejecutables.

IV – Análisis y Diseño de la Prueba

4.5.3 Desarrollo Guiado por Prueba de Aceptación (DGPA)

Ejercicio OA FL-4.5.3 :

Se le ha dado la siguiente historia:

- "Como amante de las plantas que viaja con frecuencia, quiero tener un sistema de riego automático, para que mis plantas no mueran".

También se le han dado los siguientes criterios de aceptación:

1. El agua debe abrirse cuando la temperatura sea > 30 grados y el contenido de humedad del suelo cambie de "normal" a "seco".
2. El agua debe dispersarse durante 5 minutos a razón de 15 gramos por minuto.

¿Cuál podría ser el primer caso de prueba que debe escribirse para la implementación guiada por la prueba de aceptación?

IV – Análisis y Diseño de la Prueba

4.5.3 Desarrollo Guiado por Prueba de Aceptación (DGPA)

Ejercicio OA FL-4.5.3 :

Según la historia de usuario anterior, identifique el primer caso de prueba a escribir:

a) Establezca el nivel de humedad del suelo en "mojado", luego aumente el calor y verifique que el nivel de humedad cambie a "normal" y luego a "seco" y luego a "reseco"

b) Configure la temperatura > 30 grados y verifique que la temperatura aumente en consecuencia

c) Establezca la temperatura en < 30 grados y la humedad en "seco" y verifique que no haya agua dispersa

d) Establezca la temperatura en > 30 grados y la humedad en "seco" y verifique que el agua esté dispersa

¿Debate?

IV – Análisis y Diseño de la Prueba

Final del capítulo

Actividades propuestas

- Debate:
  - ¿Coinciden los conceptos y uso de las técnicas de pruebas con las que usas en tu proyecto?
  - ¿Ves la conveniencia o necesidad de conocer técnicas de pruebas?
- Resumen. Trata de sintetizar en menos de una página los conceptos clave así como los que más dificultades te han generado
- Consulta el glosario para asegurarte que cada término utilizado lo has asimilado
- Realiza ejercicios de ejemplo de examen que corresponden a este capítulo
- Realiza consultas tanto para enfrentarte adecuadamente al examen como para aplicar los conocimientos de análisis y diseño de la prueba

Capítulo V – Gestión de las Actividades de Prueba

- V/01 Planificación de la Prueba
- V/02 Gestión del Riesgo
- V/03 Monitorización y Control de la Prueba
- V/04 Monitorización de la Prueba, Control de la Prueba y Compleción de la Prueba
- V/05 Gestión de la Configuración
- V/06 Gestión de Defectos

V – Gestión de las Actividades de Prueba

Glosario / Palabras clave en Español e Inglés

- gestión de defectos - defect management
- informe de defecto - defect report
- criterios de entrada - entry criteria
- criterios de salida - exit criteria
- riesgo de producto - product risk
- riesgo de proyecto - project risk
- riesgo - risk
- análisis del riesgo - risk análisis
- evaluación del riesgo - risk assessment
- control del riesgo - risk control
- identificación del riesgo -
- risk identification -
- nivel de riesgo - risk level
- gestión del riesgo - risk management

V – Gestión de las Actividades de Prueba

Glosario / Palabras clave en Español e Inglés

- mitigación del riesgo - risk mitigation
- monitorización del riesgo - risk monitoring
- pruebas basadas en el riesgo - risk-based testing
- enfoque de prueba - test approach
- informe de compleción de la prueba - test completion report
- control de la prueba - test control
- monitorización de la prueba - test monitoring
- plan de prueba - test plan
- planificación de prueba - test planning
- informe del avance de la prueba - test progress report
- pirámide de prueba - test pyramid
- cuadrantes de prueba - testing quadrants

V – Gestión de las Actividades de Prueba

Objetivos de Aprendizaje

5.1 Planificación de la Prueba

- FL-5.1.1 (K2) Dar ejemplos del propósito y el contenido de un plan de prueba.
- FL-5.1.2 (K1) Reconocer cómo un probador añade valor a la planificación de la iteración y de la entrega.
- FL-5.1.3 (K2) Comparar y contrastar los criterios de entrada y los criterios de salida.
- FL-5.1.4 (K3) Utilizar técnicas de estimación para calcular el esfuerzo de prueba necesario.
- FL-5.1.5 (K3) Aplicar la priorización de casos de prueba.
- FL-5.1.6 (K1) Recordar los conceptos de la pirámide de prueba.
- FL-5.1.7 (K2) Resumir los cuadrantes de prueba y sus relaciones con los niveles de prueba y los tipos de prueba.

V – Gestión de las Actividades de Prueba

Objetivos de Aprendizaje

5.2 Gestión del Riesgo

- FL-5.2.1 (K1) Identificar el nivel de riesgo utilizando la probabilidad del riesgo y el impacto del riesgo.
- FL-5.2.2 (K2) Distinguir entre riesgos de proyecto y riesgos de producto.
- FL-5.2.3 (K2) Explicar cómo el análisis del riesgo de producto puede influir en la minuciosidad y el alcance de las pruebas.
- FL-5.2.4 (K2) Explique qué medidas pueden tomarse en respuesta a los riesgos de producto analizados.

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

- FL-5.3.1 (K1) Recordar las métricas utilizadas para probar.
- FL-5.3.2 (K2) Resumir los propósitos, el contenido y las audiencias de los informes de prueba.
- FL-5.3.3 (K2) Dar ejemplos de cómo comunicar el estado de la prueba.

V – Gestión de las Actividades de Prueba

Objetivos de Aprendizaje

5.4 Gestión de la Configuración

- FL-5.4.1 (K2) Resumir cómo la gestión de la configuración apoya la prueba.

5.5. Gestión de Defectos

- FL-5.5.1 (K3) Preparar un informe de defecto.

Página intencionalmente en blanco

V – Gestión de las Actividades de Prueba

5.1.1 Propósito y Contenido de un Plan de Prueba

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

Un plan de pruebas:

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

V – Gestión de las Actividades de Prueba

5.1.1 Propósito y Contenido de un Plan de Prueba

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

V – Gestión de las Actividades de Prueba

5.1.1 Propósito y Contenido de un Plan de Prueba

El **contenido habitual de un plan de prueba** incluye: (1/2)

- **Contexto** de la prueba (por ejemplo, alcance, objetivos de la prueba, restricciones, base de prueba).
- **Supuestos y restricciones** del proyecto de prueba.
- **Implicados** (por ejemplo, roles, responsabilidades, relevancia para las pruebas, necesidades de contratación y formación).
- **Comunicación** (por ejemplo, formas y frecuencia de la comunicación, plantillas de documentación).
- Registro de **riesgos** (por ejemplo, riesgos de producto, riesgos de proyecto).

V – Gestión de las Actividades de Prueba

5.1.1 Propósito y Contenido de un Plan de Prueba

El **contenido habitual de un plan de prueba** incluye: (2/2)

- **Enfoque de prueba**
  - por ejemplo, niveles de prueba, tipos de prueba, técnicas de prueba, entregables de prueba, criterios de entrada y criterios de salida, independencia de prueba, métricas que deben recopilarse, requisitos de datos de prueba, requisitos del entorno de prueba, desviaciones de la política de prueba y la estrategia de prueba de la organización.
- **Presupuesto y calendario**.

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

V – Gestión de las Actividades de Prueba

5.1.1 Propósito y Contenido de un Plan de Prueba

Ejemplo OA FL-5.1.1.
Plan de pruebas según ISO 29119.

1. Introducción
2. Contexto del plan (objetos de prueba, alcance, restricciones, implicados)
3. Comunicación de la prueba
4. Riesgos de producto de proyecto
5. Estrategia de prueba
   1. Entregables
   2. Técnicas
   3. Criterios de compleción
   4. Métricas
   5. Entornos
   6. Pruebas de confirmación y regresión
   7. Criterio de parada y reanudación
   8. Desviaciones
6. Actividades y estimaciones
7. Dotación de personal
   1. Roles y responsabilidades
   2. Contrataciones
   3. Formación
8. Calendario

V – Gestión de las Actividades de Prueba

5.1.1 Propósito y Contenido de un Plan de Prueba

Ejemplo OA FL-5.1.1**. Plan de pruebas según IEEE 829. (Obsoleta, pero puede dar ideas)**

1. Identificador del Plan de pruebas
2. Introducción
3. Objetos de prueba
4. Características que deben ser probadas
5. Características que no deben ser probadas
6. Estrategia de pruebas
7. Criterios de aceptación y rechazo
8. Criterios para la interrupción y el reinicio de las pruebas
9. Entregables de pruebas
10. Tareas de pruebas
11. Necesidades de entorno (Infraestructura de pruebas)
12. Responsabilidades, obligaciones
13. Personal y necesidades de formación
14. Planificación / Calendario
15. Riesgos de la planificación e imprevistos
16. Aprobación y Autorización

V – Gestión de las Actividades de Prueba

5.1.2 Contribución del Probador a la planificación de Iteración y Entrega

En los CVDS **iterativos**, suelen producirse dos tipos de planificación:

- la planificación de la **entrega** y
- la planificación de la **iteración**.

5.1.2 Contribución del Probador a la planificación de Iteración y Entrega

La **planificación de la entrega**

- se anticipa al lanzamiento de un producto,
- define y redefine la lista de trabajo acumulado del producto y
- puede implicar refinar las historias de usuario más grandes en un conjunto de historias de usuario más pequeñas.

También sirve de base para el enfoque de prueba y el plan de prueba de todas las iteraciones.

- Los encargados de la planificación de la entrega
  - participan en la redacción de historias de usuario y criterios de aceptación comprobables (véase el apartado 4.5),
  - participan en los análisis de riesgos de calidad y del proyecto (véase el apartado 5.2),
  - estiman el esfuerzo de prueba asociado a las historias de usuario (véase el apartado 5.1.4),
  - determinan el enfoque de la prueba y planifican la prueba de la entrega.

V – Gestión de las Actividades de Prueba

5.1.2 Contribución del Probador a la planificación de Iteración y Entrega

La **planificación de la iteración** mira hacia el final de una única iteración y se ocupa de la lista de trabajo acumulado de la iteración.

- Los probadores implicados en la planificación de la iteración
  - participan en el análisis detallado del riesgo de las historias de usuario,
  - determinan la capacidad de ser probadas de las historias de usuario,
  - desglosan las historias de usuario en tareas (en particular, tareas de prueba),
  - estiman el esfuerzo de prueba para todas las tareas de prueba e
  - identifican y perfeccionan los aspectos funcionales y no funcionales del objeto de prueba.

5.1.3 Criterios de Entrada y Criterios de Salida

Los **criterios de entrada** definen las **precondiciones** para emprender una actividad determinada.

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

V – Gestión de las Actividades de Prueba

5.1.3 Criterios de Entrada y Criterios de Salida

Los **criterios de entrada típicos** incluyen:

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

5.1.3 Criterios de Entrada y Criterios de Salida

Los criterios de salida típicos incluyen:

- medidas de **completitud**. P. ej.,
  - nivel de cobertura alcanzado,
  - número de defectos sin resolver,
  - densidad de defectos,
  - número de casos de prueba fallidos) y
- criterios de **compleción**. P. ej.,
  - se han ejecutado las pruebas planificadas,
  - se ha realizado la prueba estática,
  - se han notificado todos los defectos encontrados,
  - se han automatizado todas las pruebas de regresión.

V – Gestión de las Actividades de Prueba

5.1.3 Criterios de Entrada y Criterios de Salida

Quedarse sin tiempo o sin presupuesto también pueden considerarse **criterios de salida válidos**.

- Incluso sin que se cumplan otros criterios de salida, puede ser aceptable finalizar las pruebas en tales circunstancias, si los implicados han revisado y aceptado el riesgo de salir a producción sin más prueba.

En el **Desarrollo Ágil de Software**,

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

5.1.3 Criterios de Entrada y Criterios de Salida

Ejemplo OA FL-5.1.3.

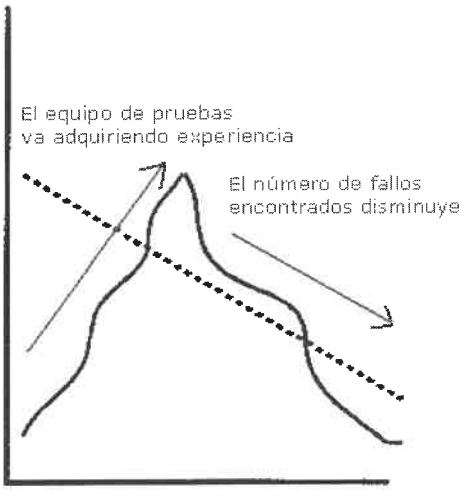
- Una configuración habitual de criterios de entrada a las pruebas (incluso en contexto poco formales) es:
  - Disponer del producto software en la versión adecuada
  - Disponer del entorno de pruebas con las características descritas
  - Disponer de las herramientas preparadas
  - Disponer de acceso a la documentación
  - Personal con cualificación adecuada y tareas asignadas
- En ocasiones se han establecido criterios de entrada muy concretos como:
  - El software se puede inicializar
  - La conexión a la BBDD es correcta
  - Se puede usar el software con un usuario y contraseña correcto
- ¿Por qué crees que se establecen criterios como estos 3 últimos?

5.1.3 Criterios de Entrada y Criterios de Salida

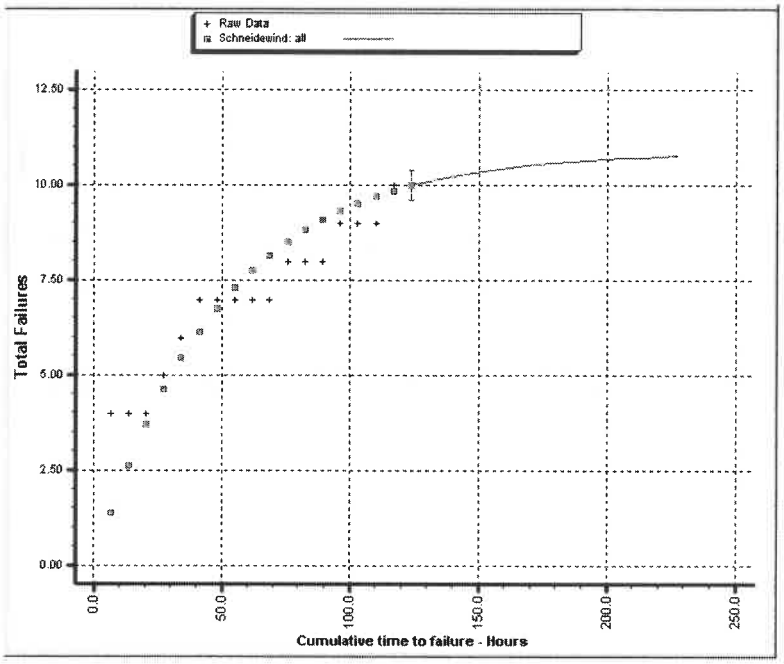
Información adicional. ¿Cuánto esfuerzo en pruebas es suficiente?

Criterios de salida

- Curvas de crecimiento de fiabilidad.
- Proporcionan un método de estimación objetiva para establecer el punto óptimo de finalización, pero requieren un control preciso del esfuerzo de prueba y un conocimiento histórico de las características del sistema.



Ejemplo OA FL-5.1.3**. Criterios de salida.**



La curva representa el número total acumulado de fallos encontrados, es muy pronunciada hasta casi el final de los 18 intervalos de pruebas, a partir de este punto el número de fallos y su severidad irá decreciendo hasta hacerse prácticamente irrelevante, momento en el que, posiblemente, se habrán identificado los defectos más relevantes y podría ser indicador del final de las pruebas. Consulte ISTQB Test Manager para mayor detalle

V – Gestión de las Actividades de Prueba
5.1.3 Criterios de Entrada y Criterios de Salida

Información adicional. ¿Cuánto esfuerzo en pruebas es suficiente?

Criterios de salida

Las técnicas de crecimiento de fiabilidad del software, se utilizan para estimar la tasa de fallos y la fiabilidad de un sistema de software durante la fase de pruebas de su desarrollo. Además, se podrá predecir el número de fallos que tendrá un sistema en la fase de producción, de forma que el gestor del proyecto dispondrá de una útil herramienta de decisión para:

1. Establecer si el conjunto de pruebas al que ha sido sometido el sistema ha resultado suficiente de forma que se pueda autorizar su paso a producción.
2. Reducir la posibilidad de que ocurra una incidencia grave en producción.
3. Estimar el momento, durante la fase de pruebas, en que se alcanzan los objetivos de fiabilidad del sistema.

5.1.4 Técnicas de Estimación

La **estimación del esfuerzo de prueba** consiste en predecir la cantidad de trabajo relacionado con la prueba que se necesita para cumplir los objetivos de un proyecto de prueba.

- Es importante dejar claro a los implicados que la estimación se basa en una serie de **suposiciones** y que siempre está sujeta a errores de estimación.
- La estimación de las tareas pequeñas suele ser más precisa que la de las grandes.
  - Por lo tanto, al estimar una tarea grande, ésta puede descomponerse en un conjunto de tareas más pequeñas que, a su vez, pueden estimarse.

Se describen las siguientes cuatro técnicas de estimación.

- Estimación basada en proporciones.
- Extrapolación.
- Delphi de Banda Ancha.
- Estimación de tres puntos.

V – Gestión de las Actividades de Prueba

5.1.4 Técnicas de Estimación

Estimación basada en proporciones.

- En esta técnica **basada en métricas**, se recopilan cifras de proyectos anteriores de la organización, lo que permite obtener proporciones "estándar" para proyectos similares.
- Las proporciones de los propios proyectos de una organización (por ejemplo, tomadas de datos históricos) suelen ser la **mejor fuente** para utilizar en el proceso de estimación.
- Estas proporciones estándar pueden utilizarse después para estimar el esfuerzo de prueba del nuevo proyecto.
- Por ejemplo OA FL-5.1.4,
  - si en el proyecto anterior la proporción del esfuerzo de desarrollo a prueba era de 3:2, y en el proyecto actual se espera que el esfuerzo de desarrollo sea de 600 días-persona, el esfuerzo de prueba puede estimarse en 400 días-persona.

5.1.4 Técnicas de Estimación

Extrapolación.

- En esta técnica basada **en métricas**, las mediciones se realizan lo antes posible en el proyecto actual para recopilar los datos.
- Al disponer de suficientes observaciones, el esfuerzo necesario para el trabajo restante puede aproximarse extrapolando estos datos (normalmente aplicando un modelo matemático).
- Este método es muy adecuado en los **CVDS iterativos**.
- Por ejemplo OA FL-5.1.4,
  - el equipo puede extrapolar el esfuerzo de prueba en la próxima iteración como el esfuerzo medio de las tres últimas iteraciones.

V – Gestión de las Actividades de Prueba

5.1.4 Técnicas de Estimación

Delphi de Banda Ancha.

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

5.1.4 Técnicas de Estimación

Estimación de tres puntos.

- En esta técnica **basada en expertos**, éstos realizan tres estimaciones:
  - la estimación más optimista (a),
  - la estimación más probable (m) y
  - la estimación más pesimista (b).
- La estimación final (E) es su **media aritmética ponderada**.
- En la versión más popular de esta técnica, la estimación se calcula como E = (a + 4*m + b) / 6.
- La ventaja de esta técnica es que permite a los expertos calcular el error de medición: SD = (b - a) / 6. **Por ejemplo** OA FL-5.1.4,
  - si las estimaciones (en horas-persona) son: a=6, m=9 y b=18, entonces la estimación final es de 10±2 horas-persona (es decir, entre 8 y 12 horas-persona), porque E = (6 + 4*9 + 18) / 6 = 10 y SD = (18 - 6) / 6 = 2.

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

V – Gestión de las Actividades de Prueba

5.1.4 Técnicas de Estimación

Ejemplo OA FL-5.1.4

Identifique las características de los siguientes escenarios de estimación:

a) Los gráficos de trabajo pendiente (burndown), que capturan e informan el esfuerzo.

b) El póquer de planificación

c) Los modelos de eliminación de defectos en el que se capturan y comunican los volúmenes de defectos y el tiempo necesario para eliminarlos

d) La técnica de estimación Delphi de Banda Ancha

5.1.4 Técnicas de Estimación

Ejemplo OA FL-5.1.4

a) Ejemplo del enfoque basado en métricas. Este dato es utilizado para alimentar la velocidad del equipo a fin de determinar la cantidad de trabajo que el equipo puede hacer en la siguiente iteración

b) Aproximación basada en expertos, dado que los miembros del equipo estiman el esfuerzo para entregar una prestación en base a su experiencia

c) Ejemplos del enfoque basado en métricas que proporciona una base para estimar futuros proyectos de naturaleza similar

d) Ejemplo de la técnica de estimación basada en expertos en la que los grupos de expertos realizan estimaciones basadas en su experiencia

5.1.5 Priorización de Casos de Prueba

Una vez que los casos de prueba y los procedimientos de prueba se han especificado y reunido en conjuntos de prueba, estos conjuntos de prueba pueden organizarse en un **calendario de ejecución de prueba** que defina el orden en que deben ejecutarse.

A la hora de priorizar los casos de prueba, pueden tenerse en cuenta distintos factores.

Las estrategias de priorización de casos de prueba más utilizadas son las siguientes:

- Priorización basada en el riesgo
- Priorización basada en la cobertura
- Priorización basada en requisitos

5.1.5 Priorización de Casos de Prueba

Priorización basada en el riesgo,

- en la que el orden de ejecución de las pruebas se basa en los resultados del análisis del riesgo (véase la sección 5.2.3).
- Los casos de prueba que cubren los riesgos más importantes se ejecutan en primer lugar.

Priorización basada en la cobertura,

- en la que el orden de ejecución de las pruebas se basa en la cobertura (por ejemplo, la cobertura de sentencias).
- Los casos de prueba que logran la mayor cobertura se ejecutan en primer lugar.

En otra variante, denominada **priorización de cobertura adicional**,

- el caso de prueba que logra la cobertura más alta se ejecuta primero;
- cada caso de prueba posterior es el que logra la cobertura adicional más alta.

V – Gestión de las Actividades de Prueba

5.1.5 Priorización de Casos de Prueba

Priorización basada en requisitos,

- en la que el orden de ejecución de las pruebas se basa en las prioridades de los requisitos trazadas hasta los casos de prueba correspondientes.
- Las prioridades de los requisitos son definidas por los implicados.
- Los casos de prueba relacionados con los requisitos más importantes se ejecutan en primer lugar.

5.1.5 Priorización de Casos de Prueba

En **condiciones ideales**, los casos de prueba se ordenarían para su realización en función de sus niveles de prioridad, utilizando, por ejemplo, una de las estrategias de priorización mencionadas.

Sin embargo, esta práctica puede no funcionar si los casos de prueba o las prestaciones que se están probando tienen **dependencias**.

Si un caso de prueba con una prioridad más alta depende de un caso de prueba con una prioridad más baja, el caso de prueba de prioridad más baja debe ejecutarse primero.

El orden de ejecución de las pruebas también debe tener en cuenta la **disponibilidad de recursos**.

- Por ejemplo, las herramientas de prueba necesarias, los entornos de prueba o las personas que sólo pueden estar disponibles durante un periodo de tiempo específico.

V – Gestión de las Actividades de Prueba

5.1.5 Priorización de Casos de Prueba

Ejercicio OA FL-5.1.5

Prioriza los casos de prueba del siguiente escenario. Un mayor valor del campo “Riego asociado” significa un mayor riesgo

Casos de Prueba	Riesgo asociado	Dependencias
CP1	5	
CP2	25	CP1
CP3	10	
CP4	20	
CP5	15	

5.1.5 Priorización de Casos de Prueba

Ejercicio OA FL-5.1.5

Calendario de ejecución de Pruebas	Riesgo asociado	Dependencias	Explicación
CP1	5		Primer caso para resolver la dependencia
CP2	25	CP1	Caso prioritario
CP4	20		2º en prioridad
CP5	15		3º en prioridad
CP3	10		4º en prioridad

5.1.6 Pirámide de Prueba

La **pirámide de prueba** es un modelo que muestra que las distintas pruebas pueden tener una granularidad diferente.

- El modelo de pirámide de prueba apoya al equipo en la automatización de la prueba y en la asignación del esfuerzo de prueba mostrando que los diferentes objetivos se apoyan en diferentes niveles de automatización de la prueba.
- Las capas de la pirámide representan grupos de pruebas.

V – Gestión de las Actividades de Prueba

5.1.6 Pirámide de Prueba

Modelo original de pirámide de prueba (Cohn 2009)

Pruebas de interfaz de usuario

Pruebas de servicio

Pruebas unitarias

V – Gestión de las Actividades de Prueba

5.1.6 Pirámide de Prueba

Otro modelo popular

Pruebas de extremo a extremo

Pruebas de integración (integración de componentes)

Pruebas unitarias (de componentes)

V – Gestión de las Actividades de Prueba

5.1.6 Pirámide de Prueba

- Cuanto **más alta es la capa**, menor es la granularidad de la prueba, el aislamiento de la prueba y el tiempo de ejecución de la prueba.
- Las pruebas de la **capa inferior** son pequeñas, aisladas, rápidas y comprueban una pequeña parte de la funcionalidad, por lo que normalmente se necesitan muchas para lograr una cobertura razonable.
- La **capa superior** representa pruebas complejas, de alto nivel y de extremo a extremo.
  - Estas pruebas de alto nivel suelen ser más lentas que las pruebas de las capas inferiores y suelen comprobar una gran parte de la funcionalidad, por lo que normalmente sólo se necesitan **unas pocas** para lograr una cobertura razonable.
- El número y la nomenclatura de las capas pueden diferir.

V – Gestión de las Actividades de Prueba

5.1.6 Pirámide de Prueba

Por **ejemplo**,

- el modelo original de pirámide de prueba (Cohn 2009) define tres capas: "pruebas unitarias", "pruebas de servicio" y "pruebas de interfaz de usuario".
- Otro modelo popular define pruebas unitarias (de componentes), pruebas de integración (integración de componentes) y pruebas de extremo a extremo.
- También pueden utilizarse otros niveles de prueba (véase la sección 2.2.1).

V – Gestión de las Actividades de Prueba

5.1.7 Cuadrantes de Prueba

Los **cuadrantes de prueba**, definidos por Brian Marick (Marick 2003, Crispin 2008), agrupan los **niveles de prueba** con los **tipos de prueba**, **actividades**, **técnicas** de prueba y **productos de trabajo** apropiados en el Desarrollo Ágil de software.

- El modelo ayuda a la gestión de pruebas a visualizarlas para asegurar que se incluyen todos los tipos de prueba y niveles de prueba apropiados en el CVDS y a comprender que algunos tipos de prueba son más relevantes para determinados niveles de prueba que otros.
- Este modelo también proporciona una forma de diferenciar y describir los tipos de pruebas a todos los implicados, incluidos desarrolladores, probadores y representantes de negocio.

V – Gestión de las Actividades de Prueba

5.1.7 Cuadrantes de Prueba

Ejemplo OA FL-5.1.7

De cara al negocio

Cuadrante Q2 • Pruebas funcionales • Ejemplos • pruebas de historias de usuario • Prototipos de experiencia de usuario • Pruebas de API y simulaciones • Comprueban los criterios de aceptación • Pueden ser **manuales** o **automatizadas**	**Cuadrante Q3** • Prueba exploratoria • Prueba de usabilidad • Prueba de aceptación de usuario • Orientadas al usuario • Suelen ser **manuales**
Cuadrante Q1 • Pruebas de componentes • Pruebas de integración de componentes. • Deberían **automatizarse** e incluirse en el proceso de IC.	**Cuadrante Q4** • Pruebas de humo • Pruebas no funcionales (excepto las pruebas de usabilidad) • **Automatizadas (con herramientas)**

Apoya al equipo

Critica al producto

Orientado a la tecnología

5.1.7 Cuadrantes de Prueba

- En este modelo, las pruebas pueden estar orientadas al negocio o a la tecnología.
- Las pruebas también pueden apoyar al equipo (es decir, guiar el desarrollo) o criticar el producto (es decir, medir su comportamiento frente a las expectativas).

La combinación de estos dos puntos de vista determina los cuatro cuadrantes: (1/2)

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

V – Gestión de las Actividades de Prueba

5.1.7 Cuadrantes de Prueba

La combinación de estos dos puntos de vista determina los cuatro cuadrantes: (2/2)

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

5.2 Gestión del Riesgo

Las organizaciones se enfrentan a muchos factores internos y externos que hacen que sea incierto si alcanzarán sus objetivos y cuándo lo harán (ISO 31000).

- La **gestión del riesgo** permite a las organizaciones aumentar la probabilidad de alcanzar los objetivos, mejorar la calidad de sus productos y aumentar la confianza de los implicados.

Las principales actividades de gestión del riesgo son:

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

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

5.2.1 Definición del Riesgo y Atributos del Riesgo

El **riesgo** es un acontecimiento, peligro, amenaza o situación potencial cuya ocurrencia causa un efecto adverso.

Un riesgo puede caracterizarse por dos factores:

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

Estos dos factores expresan el nivel de riesgo, que es una medida del riesgo.

Cuanto mayor sea el nivel de riesgo, más importante será su tratamiento.

5.2.2 Riesgos de Proyecto y Riesgos de Producto

En la prueba de software suelen preocupar dos tipos de riesgos:

- los **riesgos de proyecto** y
- los **riesgos de producto**.

V – Gestión de las Actividades de Prueba

5.2.2 Riesgos de Proyecto y Riesgos de Producto

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

Los riesgos de proyecto incluyen:

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

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

5.2.2 Riesgos de Proyecto y Riesgos de Producto

Los **riesgos de producto** están relacionados con las características de calidad del producto (por ejemplo, las descritas en el modelo de calidad ISO 25010).

Algunos ejemplos de riesgos de producto son:

- falta de funcionalidad o funcionalidad incorrecta,
- cálculos erróneos,
- errores de ejecución,
- arquitectura deficiente,
- algoritmos ineficaces,
- tiempo de respuesta inadecuado,
- mala experiencia de usuario,
- vulnerabilidades de seguridad.

V – Gestión de las Actividades de Prueba

5.2.2 Riesgos de Proyecto y Riesgos de Producto

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

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

5.2.2 Riesgos de Proyecto y Riesgos de Producto

Ejemplo OA FL-5.2.2.

- Los riesgos de proyecto son posibles sucesos que pueden influir negativamente en el buen desarrollo del proyecto de software (incluyendo sus pruebas). Podríamos decir que son eventos que pueden retrasar la entrega del software. Por ejemplo:
  - Una gripe que afecta al equipo
  - Descubrir tarde que la documentación no era suficientemente clara
- Los riesgos del producto (tanto funcionales como no funcionales) son eventos sobre dicho producto que impiden que ofrezca el servicio para el que fue construido. Por ejemplo:
  - La aplicación no me permite el acceso con un usuario válido
  - La aplicación permite el acceso a cualquiera

5.2.3 Análisis del Riesgo de Producto

Desde el punto de vista de la prueba, el **objetivo del análisis del riesgo** de producto es proporcionar una conciencia del riesgo de producto para concentrar el esfuerzo de prueba de forma que se **minimice el nivel residual de riesgo de producto**.

Lo ideal es que el análisis del riesgo de producto comience en una fase temprana del CVDS.

5.2.3 Análisis del Riesgo de Producto

El análisis del riesgo de producto consiste en

- la identificación del riesgo y
- la evaluación del riesgo.

La **identificación del riesgo** consiste en generar una lista exhaustiva de riesgos.

- Los implicados pueden identificar los riesgos utilizando diversas técnicas y herramientas, por ejemplo, tormenta de ideas, talleres, entrevistas o diagramas causa-efecto

V – Gestión de las Actividades de Prueba

5.2.3 Análisis del Riesgo de Producto

La **evaluación del riesgo** implica:

- categorizar los riesgos identificados,
- determinar su probabilidad de riesgo, su impacto y nivel de riesgo,
- establecer prioridades y
- proponer formas de gestionarlos.

La categorización ayuda a asignar acciones de mitigación, porque normalmente los riesgos que caen en la misma categoría pueden mitigarse utilizando un enfoque similar.

La evaluación del riesgo puede utilizar un enfoque cuantitativo o cualitativo, o una mezcla de ambos.

- En el **enfoque cuantitativo** el nivel de riesgo se calcula como la multiplicación de la probabilidad del riesgo y el impacto del riesgo.
- En el **enfoque cualitativo**, el nivel de riesgo puede determinarse utilizando una matriz de riesgo.

5.2.3 Análisis del Riesgo de Producto

El **análisis del riesgo de producto** puede influir en la minuciosidad y el alcance de las pruebas.

Sus resultados se utilizan para:

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

V – Gestión de las Actividades de Prueba
5.2.3 Análisis del Riesgo de Producto

Ejemplo OA FL-5.2.3.

- La experiencia nos ha mostrado que se aplican actividades de análisis de riesgos de productos en prácticamente todos los proyectos de prueba, aunque en muchos casos no se era consciente de que se hacía y no se aplicaba de la forma más eficaz y eficiente. (cientos de profesionales de pruebas, lo han comentado)
- Todos aplicamos (de mejor o peor manera) el resultado de un análisis de riegos, no solo en las pruebas, sino es nuestro ámbito personal. Por ejemplo: cuando nos vamos de vacaciones no nos llevamos “todo”, sino que seleccionamos y priorizamos
- Si se reflexiona sobre los contenidos de esta formación y sobre los conceptos de la gestión de riegos, podremos identificar un gran número de situaciones donde decidimos tareas que se hacen y tareas que no, así como la prioridad en que deben hacerse en función de los posibles problemas o beneficios a obtener. Estamos aplicando un análisis de riegos.

5.2.4 Control del Riesgo de Producto

El **control del riesgo de producto** comprende todas las medidas que se toman en respuesta a los riesgos de producto identificados y evaluados.

El control del riesgo del producto consiste en la mitigación del riesgo y la monitorización del riesgo.

- La **mitigación del riesgo** implica la implementación de las acciones propuestas en la evaluación del riesgo para reducir el nivel de riesgo.
- El objetivo de la **monitorización del riesgo** es asegurar que las acciones de mitigación son efectivas, obtener más información para mejorar la evaluación del riesgo e identificar los riesgos emergentes.

V – Gestión de las Actividades de Prueba

5.2.4 Control del Riesgo de Producto

Con respecto al control del riesgo de producto, una vez analizado el riesgo, son posibles varias opciones de respuesta al mismo, por ejemplo

- mitigación del riesgo mediante la prueba,
- aceptación del riesgo,
- transferencia del riesgo o
- plan de contingencia

(Veenendaal 2012).

5.2.4 Control del Riesgo de Producto

Las **medidas que pueden tomarse para mitigar los riesgos de producto** mediante la prueba son las siguientes:

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

V – Gestión de las Actividades de Prueba

5.2.4 Control del Riesgo de Producto

Ejemplo OA FL-5.2.4.

- Asignar las características más críticas del software donde el posible fallo puede ser muy grave, a los probadores más experimentados
- La experiencia suele recomendar más independencia de la prueba en contextos más inexpertos
- La experiencia ha demostrado que los módulos o productos más críticos deben someterse a mayor intensidad de revisión
- Las técnicas y prácticas más exhaustivas suelen aplicarse a los elementos más críticos (de mayor riesgo)

La **monitorización de la prueba** se ocupa de recopilar información sobre la prueba.

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

V – Gestión de las Actividades de Prueba
5.3 Monitorización, Control y Compleción de la Prueba

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

Algunos ejemplos de directivas de control son:

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

5.3 Monitorización, Control y Compleción de la Prueba

La **compleción de la prueba** recopila datos de las actividades de prueba finalizadas para consolidar la experiencia, el producto de prueba y cualquier otra información relevante.

Las actividades de compleción de la prueba se producen en los hitos del proyecto, como cuando

- se completa un nivel de prueba,
- se termina una iteración ágil,
- se completa (o cancela) un proyecto de prueba,
- se libera un sistema de software o
- se completa una entrega de mantenimiento.

5.3.1 Métricas Utilizadas en la Prueba

Las métricas de prueba se recopilan para mostrar

- el **avance** con respecto al calendario de prueba y el presupuesto previstos,
- la **calidad actual** del objeto de prueba y
- la **efectividad** de las actividades de prueba con respecto a los objetivos o a una meta de iteración.

La monitorización de prueba recopila una variedad de métricas para apoyar el control de prueba y la compleción de la prueba.

5.3.1 Métricas Utilizadas en la Prueba

Entre las métricas de prueba más comunes se incluyen:

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

5.3.2 Propósito, Contenido y Audiencia de los Informes de Prueba

El suministro de **información de prueba** resume y comunica la información de prueba durante y después de la prueba.

Los **informes del avance de la prueba** apoyan el control continuo de la prueba y deben proporcionar suficiente información para realizar modificaciones en el calendario de prueba, los recursos o el plan de prueba, cuando dichos cambios sean necesarios debido a una desviación del plan o a un cambio de circunstancias.

Los **informes de compleción de la prueba** resumen una etapa específica de la prueba (por ejemplo, nivel de prueba, ciclo de prueba, iteración) y pueden dar información para pruebas posteriores.

5.3.2 Propósito, Contenido y Audiencia de los Informes de Prueba

Durante la monitorización y el control de prueba, el equipo de prueba genera **informes del avance de la prueba** para los implicados con el fin de mantenerlos informados.

Los informes del avance de la prueba suelen **generarse de forma regular** (por ejemplo, a diario, semanalmente, etc.) e incluyen:

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

V – Gestión de las Actividades de Prueba

5.3.2 Propósito, Contenido y Audiencia de los Informes de Prueba

Un **informe de compleción de prueba** se prepara durante la finalización de la prueba, cuando un proyecto, nivel de prueba o tipo de prueba está completo y cuando, idealmente, se han cumplido sus criterios de salida.

Este informe utiliza los informes del avance de la prueba y otros datos.

Los informes típicos de compleción de la prueba incluyen:

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

5.3.2 Propósito, Contenido y Audiencia de los Informes de Prueba

Distintos públicos requieren información diferente en los informes e influyen en el grado de formalidad y la frecuencia de los mismos.

- Informar sobre el avance de la prueba a otras personas del mismo equipo suele ser frecuente e informal, mientras que
- informar sobre la prueba de un proyecto finalizado sigue una plantilla establecida y sólo se produce una vez.

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

V – Gestión de las Actividades de Prueba

5.3.2 Propósito, Contenido y Audiencia de los Informes de Prueba

Ejemplo OA FL-5.3.2**. Informes de pruebas**

- En el estándar 29119-3 (Documentación de la prueba) podemos encontrar una propuesta para el informe del estado de las pruebas:
  1. Resumen
  2. Información del documento (p.e. id e histórico de cambios)
  3. Introducción
     - Alcance, Referencia, Glosario
  4. Pruebas realizadas
     1. Resumen
     2. Desviaciones
     3. Evaluación
     4. Bloqueos
     5. Mediciones
     6. Riesgos
     7. Entregables
     8. Pruebas reutilizables
     9. Lecciones aprendidas

5.3.2 Propósito, Contenido y Audiencia de los Informes de Prueba

Ejemplo OA FL-5.3.2**. Informes de pruebas**

- Una adaptación más sencilla del ejemplo anterior del informe resumen de las pruebas podría ser la siguiente:
  - Identificador del documento
  - Resumen
  - Desviaciones
  - Evaluación comprensible
  - Resumen de resultados
  - Evaluación
  - Resumen de actividades

V – Gestión de las Actividades de Prueba

5.3.3 Comunicación del Estado de la Prueba

La mejor forma de comunicar el estado de la prueba **varía** en función de

- los asuntos de interés de la gestión de la prueba,
- las estrategias de prueba de la organización,
- las normas reglamentarias o,
- en el caso de los equipos autoorganizados (véase la sección 1.5.2), del propio equipo.

5.3.3 Comunicación del Estado de la Prueba

Las opciones incluyen:

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

Pueden utilizarse una o varias de estas opciones.

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

V – Gestión de las Actividades de Prueba
5.3.3 Comunicación del Estado de la Prueba

Ejemplo OA FL-5.3.3.

Algunos escenarios de comunicación de la prueba que podemos encontramos en los proyectos pueden ser:

- Enviar los resultados de la prueba en un documento ofimático
- Realizar una presentación de los resultados en una reunión de dirección
- Enviar los resultados por correo electrónico
- Disponer de un Cuadro de Mando en el entorno de integración continua
- ¿Tienes un método distinto que quieras compartir?

5.4 Gestión de la Configuración

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

Para un **elemento de la configuración complejo** (por ejemplo, un entorno de prueba), la GC registra los elementos que lo componen, sus relaciones y versiones.

Si el elemento de la configuración se aprueba para ser probado, se convierte en una **línea base** y sólo puede modificarse mediante un proceso formal de control de cambios.

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

V – Gestión de las Actividades de Prueba

5.4 Gestión de la Configuración

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

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

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

Ejemplo OA FL-5.4.1.

La GC es otro aspecto donde encaje mejor la utilización de contraejemplos, es decir, problemas derivados de una inadecuada GC

- Dificultades para localizar la documentación o información necesaria
- Utilizar una versión incorrecta de un producto o un documento
- No saber quién y porque se modificó un documento
- Ejecutar los casos de prueba inadecuados a una determinada versión del software
- Problemas con las ramas de desarrollo

Dado que uno de los principales objetivos de prueba es encontrar defectos, **es esencial contar con un proceso establecido de gestión de defectos**.

Aunque aquí nos referimos a "defectos", **las anomalías notificadas** pueden resultar ser

- defectos reales
- u otra cosa (por ejemplo,
  - un falso positivo,
  - una solicitud de cambio);

esto se resuelve durante el **proceso de tratamiento de los informes de defecto**.

Las anomalías pueden informarse durante **cualquier fase** del CVDS y la forma depende de éste.

5.5 Gestión de Defectos

Como mínimo, el proceso de gestión de defectos incluye un **flujo de trabajo** para **tratar las anomalías individuales desde su descubrimiento hasta su cierre** y reglas para su **clasificación**.

El flujo de trabajo suele incluir actividades para

- registrar las anomalías notificadas,
- analizarlas y clasificarlas,
- decidir una respuesta adecuada como arreglarlas o
- mantenerlas tal cual y, por último,
- cerrar el informe de defecto.

El proceso debe ser seguido por **todos** los implicados.

Es aconsejable tratar los **defectos de las pruebas estáticas** (especialmente el análisis estático) de forma similar.

V – Gestión de las Actividades de Prueba
5.5 Gestión de Defectos

Los **informes de defectos** típicos tienen los siguientes **objetivos**:

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

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Un informe de defecto registrado durante una prueba dinámica suele incluir: (1/2)

- Identificador único.
- Título con un breve resumen de la anomalía que se informa.
- Fecha en que se observó la anomalía, organización emisora y autor, incluido su rol.
- Identificación del objeto de prueba y del entorno de prueba.
- Contexto del defecto (por ejemplo, caso de prueba que se está realizando, actividad de prueba que se está llevando a cabo, fase del CVDS y otra información relevante como la técnica de prueba, la lista de comprobación o los datos de prueba que se están utilizando).
- Descripción del fallo para permitir su reproducción y resolución, incluidos los pasos que detectaron la anomalía, y cualquier registro de prueba, volcado de la base de datos, captura de pantalla o grabación pertinentes.
- Resultados esperados y resultados reales.

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Un informe de defecto registrado durante una prueba dinámica suele incluir: (2/2)

- Severidad del defecto (grado de impacto) sobre los intereses de los implicados o los requisitos.
- Prioridad de corrección.
- Estado del defecto (por ejemplo, abierto, aplazado, duplicado, a la espera de ser corregido, a la espera de pruebas de confirmación, reabierto, cerrado, rechazado).
- Referencias (por ejemplo, al caso de prueba).

Algunos de estos datos pueden incluirse automáticamente al utilizar **herramientas** de gestión de defectos (por ejemplo, identificador, fecha, autor y estado inicial).

En el estándar ISO/IEC/IEEE 29119-3, que se refiere a los informes de defectos como **informes de incidencias**, se pueden encontrar plantillas de documentos para un informe de defectos y ejemplos de informes de defectos.

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Ejemplo OA FL-5.5.1.

Se ha ejecutado el Caso de Prueba TC_003 para probar el requisito:

- "El sistema tendrá la opción de visualizar en pdf un listado con los nombres de todos los clientes registrados en la base de datos".

Se ha obtenido el siguiente resultado:

- "El pdf no se genera".

Teniendo en cuenta que la versión del producto probada es la 2.1 y que las pruebas se están realizando en el entorno "laboratorio", ¿cómo reportarías dicha incidencia?

Solución:

En la siguiente transparencia se muestra un ejemplo de cómo podría ser el registro de la incidencia encontrada.

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Ejemplo OA FL-5.5.1.

ID: 321

Objeto de prueba: versión 2.1

Entorno de pruebas: laboratorio

Notificador: Manuel Martín

Fecha primera aparición: 14/09/2011

Clase: MINOR

Estado: New

Prioridad: 3

Descripción breve: no se genera listado de clientes en pdf

Descripción detallada: Paso 1.- acceder al módulo de listados;
Paso 2.-seleccionar listado de clientes;
Paso 3.-pulsar el botón "view"

Caso de prueba: TC_003.

Efecto de la incidencia: las pruebas pueden continuar mientras se corrige.

Origen de la incidencia: código.

Referencia a notificaciones relacionadas: N/A.

Comentarios: N/A

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Localización de defectos durante la prueba. Información adicional.

- Los probadores ejecutan las pruebas especificadas y documentan los resultados.
- A continuación, sigue la investigación acerca de las desviaciones entre los resultados esperados y obtenidos.
  - Los efectos de los errores se hacen públicos.
- En este punto el probador ha cumplido hasta el momento con su tarea.
  - Ahora espera a someter a una repetición de las pruebas a la versión corregida.
- La gestión y supervisión posterior de la eliminación de errores es asumida a partir de este punto por la gestión de incidencias.

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Tareas de la gestión de defectos / incidencias. Información adicional

- En la gestión de incidencias se recogen todas las incidencias / errores encontrados en el proyecto.
- Además de su recogida y administración se establecen informaciones adicionales para los errores individuales.
  - Se determina la gravedad del error.
  - Estado del error- p.ej., abierto, en proceso de trabajo, solucionado…
- Todas las informaciones se almacenan en una base de datos central.
  - Aquí es posible un tratamiento sin redundancias.
  - Se proporciona una visión óptima acerca de los errores encontrados y su tratamiento.

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Tareas de la gestión de defectos / incidencias. Información adicional

- Sin un proceso de gestión de incidencias en funcionamiento no se puede tener un tratamiento ordenado de las mismas.
  - La gestión de incidencias debe construirse al principio del proyecto.
- Algunos puntos que deben ser considerados para una gestión correcta de incidencias son:
  - Proporcionar notificaciones / formularios de notificación comunes.
  - Todos las incidencias / errores / problemas deben ser recogidos, independientemente de parte de qué persona involucrada en el proceso provengan y de dónde hayan aparecido.
  - Se debe definir una manera de proceder determinada para el tratamiento de las incidencias (definir el proceso).
  - Debe de existir en la organización unas normas establecidas para clasificar las incidencias.

V – Gestión de las Actividades de Prueba

5.5 Gestión de Defectos

Reparto de tareas. Información adicional.

- Probador.
  - Ejecuta las pruebas para detectar errores.
  - Documenta los resultados (Documentación de pruebas).
  - Registra los errores en la base de datos (Notificación de errores y desviaciones).
- Gestor de pruebas.
  - Evalúa las notificaciones de errores.
  - Otorga prioridades (de acuerdo con el director de proyecto, el cliente, ...)
  - Supervisa el tratamiento.
- Gestión de las modificaciones o Change Control Board (CCB)
  - Toma decisiones sobre las modificaciones y su priorización.
- Desarrollador.
  - Corrige los errores de acuerdo con sus prioridades.
  - Lleva a cabo todas las modificaciones acordadas.
- Las tareas se resuelven en un proceso iterativo:
  - Probador
  - Gestor de pruebas
  - Gestión de las modificaciones (o CCB)
  - Desarrollador

Información adicional.

Clases

- La gravedad de una incidencia / error se determina según su partición:
  - Las clases se pueden formar libremente.
  - Las clases proporcionan diferentes niveles que se puede asignar a una incidencia (críticas, graves, medias, ligeras).
  - La base para la clasificación puede ser el grado de perjuicio en la implantación del producto [DIN 66271].

Priorización

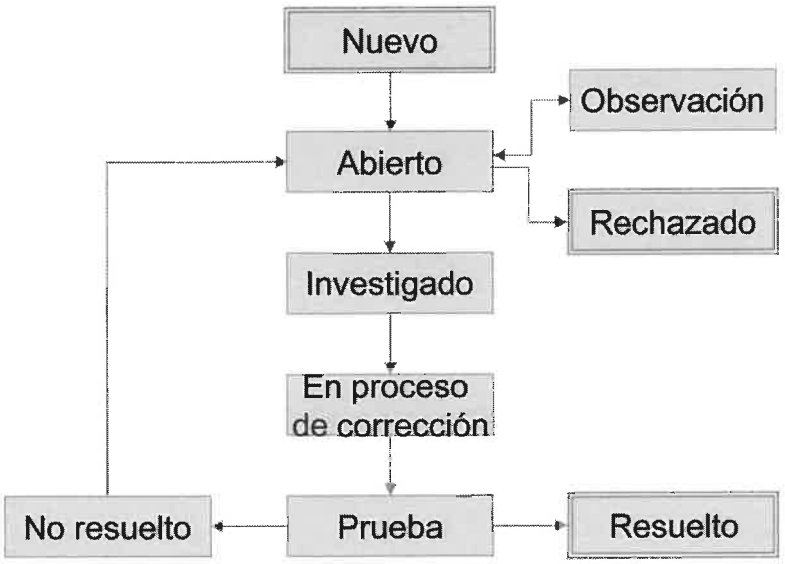
- El tratamiento / la corrección de errores se lleva a cabo en función de la prioridad asignada:
  - Un criterio de priorización en sin duda la gravedad del error, en la que se tienen en cuenta los efectos del mismo.
  - Otros criterios pueden ser también el desarrollo del proyecto, etc. ¿En que medida se impide el desarrollo posterior del proceso?
  - Las prioridades podrían ser: eliminación indispensable, eliminación en el siguiente tratamiento posterior del objeto, etc.

Estado del defecto / incidencia. Información adicional.

- El estado de la incidencia determina en que fase de elaboración se encuentra.
- Posibles caracterizaciones pueden ser:
  - **Nuevo** – el error o incidencia ha sido notificado por primera vez por parte del probador.
  - **Abierto** – la notificación se liberó por parte del gestor de pruebas.
  - **Rechazado** – la notificación fue rechazada por el gestor de pruebas.
  - **Investigado** – el desarrollador intenta identificar el error.
  - **Observación** – no se puede identificar el error, está en observación.
  - **En proceso** – el error se ha liberado para su corrección (en caso contrario se rechaza).
  - **Repetición de la prueba** –se asigna por parte del desarrollador después de la corrección.
  - **Solucionado** – se asigna por el probador cuando el error se demuestra eliminado después de la repetición de la prueba.
  - **No solucionado** – la asigna el probador cuando el error continúa apareciendo.

Ejemplo de ciclo de vida de un defecto / incidencia. Información adicional.

- Se pueden dar las siguientes transiciones de estado:



V – Gestión de las Actividades de Prueba
5.5 Gestión de Defectos

Estados. Información adicional.

- Sólo un probador puede confirmar un error como solucionado.
- El gestor de pruebas o el gestor de modificaciones decide si una incidencia debe ser rechazada o trabajada en base a las informaciones disponibles acerca del suceso y los costes de su eliminación.
- Todas las modificaciones junto a los comentarios se recogen en la gestión de incidencias.
  - Se proporciona un control constante sobre su tratamiento.
  - Se puede determinar la continuación de las pruebas.
  - En su caso se deben generar nuevos casos de prueba si la incidencia en cuestión no surgió durante la comprobación mediante casos de prueba.

V – Gestión de las Actividades de Prueba

Final del capítulo

Actividades propuestas

- Debate:
  - ¿Coinciden los conceptos y uso de las técnicas de gestión de pruebas con las que usas en tu proyecto?
  - ¿Ves la conveniencia o necesidad de gestionar las pruebas?
- Resumen. Trata de sintetizar en menos de una página los conceptos clave, así como los que más dificultades te han generado
- Consulta el glosario para asegurarte que cada término utilizado lo has asimilado
- Realiza ejercicios de ejemplo de examen que corresponden a este capítulo
- Realiza consultas tanto para enfrentarte adecuadamente al examen como para aplicar los conocimientos a gestión de las pruebas

Capítulo VI – Herramientas de Prueba

- VI/01 Herramientas de Apoyo a la Prueba
- VI/02 Ventajas y Riesgos de la Automatización de la Prueba

VI – Herramientas de Prueba

Glosario / Palabras clave en Español e Inglés

- automatización de la prueba - test automation

VI – Herramientas de Prueba

Objetivos de Aprendizaje

6.1 Herramientas de Apoyo a la Prueba

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

6.2 Ventajas y Riesgos de la Automatización de la Prueba

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

VI – Herramientas de Prueba

6.1 Herramientas de Apoyo a la Prueba

Las herramientas de prueba apoyan y facilitan muchas actividades de prueba. Algunos ejemplos son (1/2)

- Herramientas de **gestión** - aumentan la eficiencia del proceso de prueba facilitando la gestión del CVDS, los requisitos, las pruebas, los defectos y la configuración.
- Herramientas de **prueba estática**: ayudan al probador a realizar revisiones y análisis estáticos.
- Herramientas de **diseño de pruebas e implementación**: facilitan la generación de casos de prueba, datos de prueba y procedimientos de prueba.
- Herramientas de **ejecución de la prueba y de cobertura**: facilitan la ejecución automatizada de la prueba y la medición de la cobertura.
- Herramientas de **pruebas no funcionales** - permiten al probador realizar pruebas no funcionales que son difíciles o imposibles de realizar manualmente

VI – Herramientas de Prueba

6.1 Herramientas de Apoyo a la Prueba

Las herramientas de prueba apoyan y facilitan muchas actividades de prueba. Algunos ejemplos son (2/2)

- Herramientas **DevOps** - apoyan la canalización de entrega DevOps, el seguimiento del flujo de trabajo, los procesos de construcción automatizados, IC/EC
- Herramientas de **colaboración** - facilitan la comunicación
- Herramientas de apoyo a la **escalabilidad** y la normalización del **despliegue** (por ejemplo, máquinas virtuales, herramientas de contenerización)
- **Cualquier otra herramienta** que ayude en las pruebas (por ejemplo, una hoja de cálculo es una herramienta de prueba en el contexto de las pruebas)

VI – Herramientas de Prueba

6.1 Herramientas de Apoyo a la Prueba

Ejemplo OA FL-6.1.1 .

Con el fin de conocer los distintos tipos de herramientas que se usan en las empresas, vamos a realizar la siguiente dinámica:

- Por cada uno de los grupos de actividades relacionadas con las pruebas que se han mostrado indica:
  - ¿Qué herramienta usas?
  - ¿Qué aspectos has tenido que personalizar de dicha herramienta?
  - ¿Qué dificultades te has encontrado?

La mera adquisición de una herramienta no garantiza el éxito.

Cada nueva herramienta requerirá un esfuerzo para lograr beneficios reales y duraderos (por ejemplo, para la introducción de la herramienta, el mantenimiento y la formación).

También existen algunos riesgos, que necesitan análisis y mitigación.

VI – Herramientas de Prueba
6.2 Ventajas y Riesgos de la Automatización de la Prueba

Entre los **beneficios** potenciales de utilizar la automatización de la prueba se incluyen: (1/2)

- **Ahorro de tiempo** al reducir el trabajo manual repetitivo (por ejemplo, ejecutar pruebas de regresión, volver a introducir los mismos datos de prueba, comparar los resultados esperados frente a los resultados reales y cotejarlos con los estándares de codificación).
- **Prevención de errores humanos** simples gracias a una mayor consistencia y repetibilidad (por ejemplo, las pruebas se obtienen de forma coherente a partir de los requisitos, los datos de prueba se crean de forma sistemática y las pruebas se ejecutan mediante una herramienta en el mismo orden y con la misma frecuencia).
- **Evaluación más objetiva** (por ejemplo, la cobertura) y proporcionar medidas que son demasiado complicadas de obtener para los humanos

VI – Herramientas de Prueba

6.2 Ventajas y Riesgos de la Automatización de la Prueba

Entre los **beneficios** potenciales de utilizar la automatización de la prueba se incluyen: (2/2)

- **Acceso más fácil a la información** sobre pruebas para apoyar la gestión de pruebas y la elaboración de informes de pruebas (por ejemplo, estadísticas, gráficos y datos agregados sobre el avance de las pruebas, las tasas de defectos y la duración de la ejecución de las pruebas).
- **Tiempos de ejecución de pruebas reducidos** para proporcionar una detección de defectos más temprana, una retroalimentación más rápida y un tiempo de comercialización más rápido
- **Más tiempo para que los probadores** diseñen pruebas nuevas, más profundas y más eficaces

VI – Herramientas de Prueba

6.2 Ventajas y Riesgos de la Automatización de la Prueba

Entre los **riesgos** potenciales de utilizar la automatización de la prueba se incluyen: (1/2)

- **Expectativas poco realistas** sobre las ventajas de una herramienta (incluidas la funcionalidad y la facilidad de uso).
- **Estimaciones imprecisas** del tiempo, los costes y el esfuerzo necesarios para introducir una herramienta, mantener los guiones de prueba y cambiar el proceso de prueba manual existente.
- Utilizar una herramienta de prueba cuando **la prueba manual es más apropiada**.
- **Confiar demasiado en una herramienta**, por ejemplo, ignorando la necesidad del pensamiento crítico humano.
- La **dependencia del proveedor de la herramienta**, que puede quebrar, retirar la herramienta, venderla a otro proveedor o proporcionar un soporte deficiente (por ejemplo, respuestas a consultas, actualizaciones y corrección de defectos).

VI – Herramientas de Prueba

6.2 Ventajas y Riesgos de la Automatización de la Prueba

Entre los **riesgos** potenciales de utilizar la automatización de la prueba se incluyen: (2/2)

- Utilizar un **software de código abierto que puede estar abandonado**, lo que significa que no hay más actualizaciones disponibles, o sus componentes internos pueden requerir actualizaciones bastante frecuentes como desarrollo posterior.
- La **herramienta de automatización no es compatible** con la plataforma de desarrollo.
- **Elección de una herramienta inadecuada** que no cumpla los requisitos reglamentarios y/o los estándares de seguridad física.

VI – Herramientas de Prueba

6.2 Ventajas y Riesgos de la Automatización de la Prueba

Ejemplo:

Con el fin de conocer los beneficios y problemas con las herramientas que se usan en las empresas, vamos a realizar la siguiente dinámica

- Indica algún suceso relevante con la instalación y uso de herramientas relacionadas con las pruebas, bien sea como un beneficio obtenido como un problema sufrido
- Nota: Es posible que los sucesos reales expuestos por el profesor sean mayoritariamente de problemas
  - Herramientas que se infrautilizaron
  - Herramientas que se abandonaron por sobrecostes no previstos
  - Situaciones de "es que a mano lo hago antes" ☹
  - Frustración porque la herramienta no aportó lo que se esperaba

VI – Herramientas de Prueba

Final del capítulo

Actividades propuestas

- Debate:
  - ¿Coinciden la visión y enfoque de las herramientas de prueba con las que aplicas en tu proyecto?
  - ¿Ves la conveniencia o necesidad de gestionar las herramientas de prueba?
- Resumen. Trata de sintetizar en menos de una página los conceptos clave, así como los que más dificultades te han generado
- Consulta el glosario para asegurarte que cada término utilizado lo has asimilado
- Realiza ejercicios de ejemplo de examen que corresponden a este capítulo
- Realiza consultas tanto para enfrentarte adecuadamente al examen como para aplicar los conocimientos a tu automatización de la prueba y tus herramientas

Capítulo VII – Referencias

- VII/01 Estándares
- VII/02 Libros, Artículos y Páginas Web

VII – Referencias

Estándares

- **ISO/IEC/IEEE 29119-1** (2022) Software and systems engineering – Software testing – Part 1: General Concepts
- **ISO/IEC/IEEE 29119-2** (2021) Software and systems engineering – Software testing – Part 2: Test processes
- **ISO/IEC/IEEE 29119-3** (2021) Software and systems engineering – Software testing – Part 3: Test documentation
- **ISO/IEC/IEEE 29119-4** (2021) Software and systems engineering – Software testing – Part 4: Test techniques
- **ISO/IEC 25010**, (2011) Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) System and software quality models
- **ISO/IEC 20246** (2017) Software and systems engineering – Work product reviews
- **ISO/IEC/IEEE 14764:2022** – Software engineering – Software life cycle processes – Maintenance ISO 31000 (2018) Risk management – Principles and guidelines

VII – Referencias

Libros, Artículos y Páginas Web

Consulte el programa de estudios para obtener una relación de los libros, artículos y páginas web utilizados para desarrollar los contenidos

Les agradecemos el interés mostrado

y

les deseamos una evaluación final exitosa
para el
"Probador Certificado – Nivel Básico"

ANEXOS

Anexos

Nota

Sobre la información contenida en los anexos de la documentación

- La información contenida en los anexos no forma parte del programa de estudios y por tanto está fuera del alcance del examen de certificación
- Sin embargo, dichos contenidos son interesantes para varios objetivos
  - Ver algunos conceptos con otra descripción, lo que puede ayudar a la comprensión de los contenidos
  - Se presentan situaciones con cierta similitud a la realidad que pueden ayudar a comprender a aplicación del concepto y servir de ejemplos
  - Amplían el conocimiento de la materia descrita más allá de los estrictamente necesarios para realizar el examen

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Componente. Información adicional

Objetivos de la Prueba de Componente

- La prueba de componente (también conocida como **prueba unitaria** o de **módulo**) se centra en los componentes que se pueden probar por separado
- Objetivos de la prueba de componente:
  - Reducir el riesgo.
  - Verificar que los comportamientos funcionales y no funcionales del componente son los diseñados y especificados.
  - Generar confianza en la calidad del componente.
  - Encontrar defectos en el componente (internos).
  - Prevenir la propagación de defectos a niveles de prueba superiores.
- En desarrollos donde los cambios son continuos (incrementales e iterativos, Ágiles), la prueba de regresión de componente automatizada es clave para generar la confianza de que los cambios en un componente no dañan a otros componentes existentes
- Los efectos hacia otros componentes del programa quedan fuera de consideración.

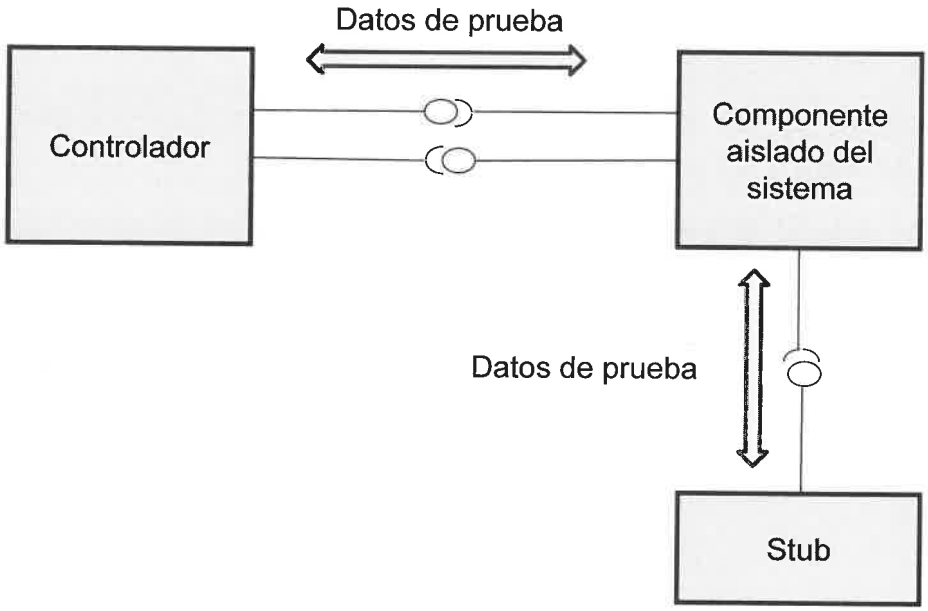
II – Prueba a lo largo del CVDS

2.2.1 Prueba de Componente. Información adicional

Objetivos de la Prueba de Componente

- La prueba de componente se realiza habitualmente de forma aislada del resto del sistema, aunque depende del modelo de ciclo de vida de software
- Esta prueba aislada puede requerir:
  - Objetos simulados
  - Virtualización de servicios
  - Arneses
  - Stubs
  - Controladores
- La prueba de componente puede cubrir:
  - la funcionalidad (por ejemplo, la exactitud de los cálculos)
  - las características no funcionales (por ejemplo, la búsqueda de fugas de memoria)
  - las propiedades estructurales (por ejemplo, pruebas de decisión).

Objetivos de la Prueba de Componente



II – Prueba a lo largo del CVDS
2.2.1 Prueba de Componente. Información adicional

Objetivos de la Prueba de Componente. Información adicional

- El **controlador** es un componente o herramienta que sustituye temporalmente a otro componente y controla o llama a un elemento de prueba aislado (componente)
- Los controladores de pruebas, junto con los stubs, posibilitan la ejecución de los componentes en el entorno del software
  - Los controladores de pruebas simulan entradas, p.ej., del usuario
  - Los controladores de pruebas recogen valores de salida
- Opcionalmente un controlador de pruebas también permite el registro de diferentes valores.
- Los controladores de pruebas pueden ser programados por uno mismo si:
  - Se tiene conocimientos de programación.
  - El código de programa está disponible.
  - Se dispone de herramientas de programación

2.2.1 Prueba de Componente. Información adicional

Objetivos de la Prueba de Componente. Información adicional

- Los **stubs** se necesitan cuando los componentes a probar requieren por su parte entradas o deben manipular salidas. Reemplazan o simulan componentes que aún no están disponibles.
  - Los stubs no contienen a diferencia de los controladores ninguna o una muy pequeña lógica de programación.
  - Los stubs proporcionan en ocasiones datos de pruebas (Dummies).

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Componente. Información adicional

Objetivos de la Prueba de Componente. Información adicional

- **Pruebas de robustez**
- Se prueba cómo de robusto es un componente.
  - La robustez describe la capacidad de respuesta de un software frente a entradas invalidas, etc.
- La robustez se prueba mediante casos de prueba negativos.
  - Los casos de prueba negativos son casos que contienen datos de entrada incorrectos o no permitidos.
- Si el componente dispone de un tratamiento de excepciones para cada posible entrada de datos errónea, se considera robusto.
  - Sin el correspondiente tratamiento de excepción los datos erróneos se introducen en el proceso y pueden ocasionar errores.
  - Las posibles consecuencias son caídas y mal funcionamiento de los componentes

2.2.1 Prueba de Componente. Información adicional

Bases de Prueba:

- Diseño detallado
- Código
- Modelo de datos
- Especificación del componente

Objetos de prueba:

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

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Componente. Información adicional

Defectos y Fallos Característicos

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

- Por lo general, los defectos se corrigen tan pronto como se detectan
- A menudo sin una gestión formal de los defectos
- Cuando los desarrolladores informan sobre defectos, esto proporciona información importante para el análisis de la causa raíz y la mejora del proceso.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Componente. Información adicional

Enfoques y Responsabilidades Específicos

- En general, el desarrollador que escribió el código realiza la prueba de componente
- Los desarrolladores pueden alternar el desarrollo de componentes con la búsqueda y corrección de defectos
- A menudo los desarrolladores escriben y ejecutan pruebas después de haber escrito el código de un componente
- Especialmente el desarrollo ágil, los casos de prueba de componente automatizados se crean antes que la creación del código

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Componente. Información adicional

Enfoques y Responsabilidades Específicos. Información adicional

- El probador dispone del código de programa de los componentes
  - Si Probador=Desarrollador (habitual):
    se puede probar de forma muy cercana al desarrollo
  - El conocimiento acerca de funciones, estructuras y variables pueden ser utilizados para la generación de casos de prueba (caja blanca)
  - A menudo se utilizan pruebas funcionales.
  - Mediante herramientas (depuradores) se posibilita la observación y actuación sobre las variables del programa.
- El conocimiento de código fuente posibilita, en general, la utilización de procedimientos de caja blanca para las pruebas de componentes.

2.2.1 Prueba de Integración. Información adicional

Objetivos de la Prueba de Integración

- La prueba de integración se centra en las interacciones entre componentes o sistemas
- Objetivos de la prueba de integración:
  - Reducir el riesgo.
  - Verificar que los comportamientos funcionales y no funcionales de las interfaces sean los diseñados y especificados.
  - Generar confianza en la calidad de las interfaces.
  - Encontrar defectos (que pueden estar en las propias interfaces o dentro de los componentes o sistemas).
  - Prevenir la propagación de defectos a niveles de prueba superiores.
- De la misma forma que con la prueba de componente, en algunos casos la prueba de regresión de integración automatizada proporciona la confianza de que los cambios no han dañado a las interfaces, los componentes o los sistemas existentes
- **Cada elemento básico debería haber sido comprobado antes de que comience la integración**

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Objetivos de la Prueba de Integración

Hay 2 niveles de prueba de integración según el tamaño del objeto de prueba

- **Prueba de integración de componentes**
  - Se centra en las interacciones e interfaces entre los componentes integrados
  - Se realiza después de la prueba de componente
  - En general, está automatizada
  - En el desarrollo iterativo e incremental suele formar parte del proceso de integración continua
- **Prueba de integración de sistemas**
  - Se centra en las interacciones e interfaces entre sistemas, paquetes y microservicios
  - Puede cubrir las interacciones con interfaces proporcionadas por organizaciones externas (por ejemplo, servicios web)
  - Puede realizarse después de la prueba de sistema o en paralelo con las actividades de prueba de sistema en curso (tanto en el desarrollo secuencial como en el desarrollo iterativo e incremental)

2.2.1 Prueba de Integración. Información adicional

Bases de Prueba:

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

Objetos de prueba:

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

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Defectos y Fallos Característicos

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

Defectos y Fallos Característicos

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

II – Prueba a lo largo del CVDS
2.2.1 Prueba de Integración. Información adicional

Enfoques y Responsabilidades Específicos

- La prueba de integración (tanto componentes como sistemas) debe concentrase en la integración propiamente dicha
- La prueba debe concentrarse en la comunicación entre módulos (o sistemas), no en la funcionalidad individual de los módulos (o sistemas)
- Se puede utilizar los tipos de prueba funcional, no funcional y estructural
- La prueba de integración de componentes suele ser responsabilidad de los desarrolladores
- La prueba de integración de sistemas es, en general, responsabilidad de los probadores
- En condiciones ideales, los probadores que realizan la prueba de integración de sistemas deberían entender la arquitectura del sistema y deberían haber influido en la planificación de la integración

2.2.1 Prueba de Integración. Información adicional

Enfoques y Responsabilidades Específicos

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

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Enfoques y Responsabilidades Específicos

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

- Comentario (sobre la psicología de las pruebas): En las pruebas de integración debería tenerse en cuenta que existe la posibilidad de tener que reunir los resultados de diferentes desarrolladores y/o equipos de pruebas.

2.2.1 Prueba de Integración. Información adicional

Enfoques y Responsabilidades Específicos. Información adicional

- Se necesitan controladores de pruebas (que proporcionan el entorno de ejecución del sistema / subsistema) que:
  - Produzcan/posibiliten entradas y salidas para el sistema / sub-sistema.
  - Registren datos.
- Aquí pueden ser reutilizados los controladores de casos de pruebas utilizados en las pruebas de componentes.
- En este punto se pueden introducir con sentido **monitores** que permitan un control de las pruebas mediante la captura de datos.
- Los stubs sustituyen a los componentes que faltan:
  - Los datos/funciones de un componente que no hayan sido integrados se introducen mediante stubs especialmente programados.
  - Los stubs asumen las tareas elementales de los componentes que falten.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Enfoques y Responsabilidades Específicos. Información adicional

- Los errores de interfaz no aparecen hasta después de la integración de los elementos básicos:
  - Si se cambian los controladores de pruebas y los stubs por interfases de componentes "reales" pueden aparecer circunstancialmente errores.
  - P. ej. se pueden perder ciertos datos, no transmitirse o transmitirse de manera errónea, etc.
- Las pruebas de integración deberían asentarse sobre las pruebas de componentes:
  - Todos los errores de las pruebas de componentes podrían ser encontrados, en principio, en las pruebas de integración, pero sería muy complicado localizarlos.
  - Probar los componentes en el nivel de integración es confuso.
  - No todos los casos de prueba pueden siquiera "lanzarse".

Estrategias de integración. Información adicional

- Existen diferentes alternativas para las estrategias de integración.
  - Generalmente las estrategias serán establecidas por el gestor de pruebas.
  - El enfoque incremental es común a todas las estrategias (excepción: Big Bang).
- La elección de la estrategia deberá tener en cuenta la eficiencia de las pruebas
  - La estrategia de integración decide acerca del esfuerzo de las pruebas (p.ej. utilización de herramientas, programación de controladores de pruebas, stubs etc.)
  - La terminación de los componentes fija para cada estrategia de integración el marco temporal.
- Sin unos componentes probados no es eficaz una estrategia de integración encaminada a ahorrar esfuerzos.
- En función del proyecto se deberá sopesar entre el ahorro de tiempo y el esfuerzo para las pruebas:
  - **Probar lo que está terminado** – eventualmente mayor esfuerzo, por el contrario, no existen periodos de inactividad.
  - **Seguir una estrategia establecida** – menor esfuerzo pero, por el contrario, eventualmente, periodos de inactividad.

II – Prueba a lo largo del CVDS
2.2.1 Prueba de Integración. Información adicional

Estrategias de integración. Información adicional

- **Estrategia Ascendente** (Bottom-Up):
  - Se lleva a cabo una integración de abajo a arriba.
  - Se empieza por aquellos componentes que no llaman a otras partes del programa.
  - Siguen paso a paso los componentes que sólo llaman a la parte ya integrada.
  - Se aplica en desarrollos nuevos, pruebas de equipos grandes y distribuidos.
- Características de la estrategia Bottom-Up:
  - No existen huecos que deban ser completados por stubs.
  - Los componentes superiores deben ser simulados mediante controladores de pruebas.
- Implantación de la estrategia Bottom-Up:
  - La implantación de la estrategia sólo es posible para software construido jerárquicamente de manera constante – por ello en su forma pura no tiene apenas relevancia práctica.

2.2.1 Prueba de Integración. Información adicional

Estrategias de integración. Información adicional

- **Estrategia Descendente** (Top-Down):
  - La integración se lleva a cabo sistemáticamente de arriba a abajo.
  - Se empieza por los componentes que no son llamadas desde otros componentes del software.
  - Paso a paso se integran aquellos componentes que únicamente son llamados desde la parte ya integrada.
  - Se aplica cuando se utiliza software ajeno o Frameworks.
- Características de la estrategia Top-Down:
  - Debido a la forma de proceder no son necesarios controladores de pruebas.
  - Todos los componentes subordinados deben ser sustituidos por stubs.
- Implantación de la estrategia Top-Down:
  - La implantación de la estrategia sólo es posible para software construido jerárquicamente de manera continuada – por ello en su forma pura no tiene apenas relevancia práctica.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Estrategias de integración. Información adicional

- Integración orientada al **proceso de negocio**:
  - La integración se lleva a cabo por procesos de negocio.
  - En cada caso se integran los componentes necesarios por parte de un proceso de negocio.
  - También se denominan como pruebas End-to-End.
  - Características de la integración orientada al proceso de negocio:
  - Los controladores de pruebas sustituyen los componentes de rango superior.
  - Todos los componentes subordinados de deben sustituir por stubs.
- Integración orientada a **funciones**:
  - La integración se orienta a una función del sistema escogida.
  - Se integra cada uno de los componentes necesitados por la función correspondiente del sistema.
- Características de la integración orientada a funciones:
  - Los controladores de pruebas sustituyen a los componentes de rango superior.
  - Todos los componentes subordinados deben ser sustituidos mediante stubs

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Estrategias de integración. Información adicional

- Integración Ad-Hoc:
  - Los componentes ya probados son integrados – en tanto sea posible- inmediatamente después de su programación y comprobación exitosa.
- Características de la integración Ad-Hoc:
  - No existen retrasos en las pruebas, por lo que ocasionalmente se puede lograr un acortamiento del proceso completo de desarrollo del software.
  - Dependiendo del tipo de componente terminado pueden ser necesarios stubs y controladores de pruebas.
- Implantación de la estrategia Ad-Hoc:
  - La integración Ad-Hoc puede ser empleada en cualquier situación – en la práctica suele ser combinada con otras estrategias.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Estrategias de integración. Información adicional

- Integración **no incremental** (big bang):
  - Se integra en el momento en que se dispone de todos los componentes.
  - Esta estrategia se aplica, p. ej., en proyectos de mantenimiento o ampliación, así como en el marco de migraciones.
- Características de la integración no incremental:
  - El área de las pruebas tiene mucho "tiempo muerto" – no se puede probar durante largo tiempo.
  - Las pruebas se hacen más complicadas y más amplias - los efectos de los errores aparecen con frecuencia y son difíciles de rastrear.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Integración. Información adicional

Estrategias de integración. Información adicional

- **¿Como se pueden coordinar la integración y las pruebas?**

La integración debe llevarse a cabo de acuerdo con la arquitectura del sistema y la preparación de los componentes.

Las pruebas deben ser eficientes, es decir, encontrar el mayor número de errores graves sin utilizar para ello demasiados recursos.

Se dispone de una estrategia de integración que sirve también de base para el proceso de pruebas.

Determina también el esfuerzo de las pruebas pero depende de la disponibilidad de los componentes.

En la planificación de las pruebas se determinan el tiempo y el resto de recursos para las pruebas – ¡si no se puede probar según el plan aparecen costes adicionales!

En principio el orden de la preparación determina la integración - y de esta manera también el curso de las pruebas .

Se debe componer una estrategia que se adapte individualmente.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Sistema. Información adicional

Objetivos de la Prueba de Sistema

La prueba de sistema se centra en el comportamiento y capacidades de todo el sistema o producto. Tiene en cuenta:

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

2.2.1 Prueba de Sistema. Información adicional

Objetivos de la Prueba de Sistema

- Para ciertos sistemas, es relevante el objetivo de la calidad de los datos
- La prueba de regresión de sistema automatizada proporciona confianza de que los cambios no han dañado características existentes o capacidades extremo a extremo (al igual que en niveles de componente e integración)
- Habitualmente, la prueba de sistema aporta información que es utilizada por los implicados para la toma de decisiones respecto al lanzamiento del producto
- La prueba de sistema puede satisfacer requisitos o estándares legales o regulatorios
- El entorno de prueba debe corresponder, en condiciones ideales, al entorno objetivo final o entorno de producción.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Sistema. Información adicional

Bases de Prueba. Ejemplos

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

2.2.1 Prueba de Sistema. Información adicional

Objetos de Prueba característicos

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

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Sistema. Información adicional

Defectos y Fallos Característicos. Ejemplos

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

Enfoques y Responsabilidades Específicos

- La prueba de sistema debe centrarse en el comportamiento global y extremo a extremo del sistema en su conjunto, tanto funcional como no funcional.
- Deben utilizarse las técnicas más apropiadas (ver capítulo 4).
  - Ejemplo, se puede crear una tabla de decisión para verificar si el comportamiento funcional es el descrito en términos de reglas de negocio.
- En general, la prueba de sistema se realiza por probadores independientes
- Los defectos en las especificaciones (p.ej., falta de historias de usuario, requisitos de negocio establecidos de forma incorrecta, etc.) pueden llevar a una falta de comprensión o a desacuerdos sobre el comportamiento esperado del sistema. Tales situaciones pueden causar:
  - falsos positivos: que hace perder tiempo
  - falsos negativos: que reduce la eficacia de la detección de defectos
- La participación temprana de los probadores en el perfeccionamiento de la historia de usuario o en actividades de prueba estática, como las revisiones, ayuda a reducir la incidencia de tales situaciones.

II – Prueba a lo largo del CVDS
2.2.1 Prueba de Sistema. Información adicional

Información adicional

- Desde el punto de vista técnico ya se han probado todos los componentes y su interrelación
- Son pruebas del sistema completo en condiciones de funcionamiento y desde el punto de vista del usuario (entorno, funciones, carga)
- Son pruebas del sistema integrado desde el punto de vista del usuario
  - Implantación completa y correcta de los requisitos.
  - Implantación en el entorno real del sistema y con datos cercanos a la práctica.
- El entorno de pruebas debería corresponderse con el entorno de producción
  - Se omiten los controladores de pruebas y los *stubs.*
  - Todas las interfaces externas del sistema se probarán bajo condiciones de producción.
  - Recreación lo más exacta posible del entorno posterior de producción.
- Pero: ninguna prueba en el entorno de producción
  - Los errores surgidos pueden dañar el sistema productivo.
  - El entorno del sistema está en movimiento - las pruebas no son reproducibles.

2.2.1 Prueba de Aceptación. Información adicional

Objetivos de la Prueba de Aceptación

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

Los objetivos de la prueba de aceptación son:

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

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Aceptación. Información adicional

Objetivos de la Prueba de Aceptación

La prueba de aceptación puede producir información para evaluar el grado de preparación del sistema para su despliegue y uso por el cliente (usuario final)

Los defectos pueden encontrarse durante la prueba de aceptación, pero encontrar defectos no suele ser un objetivo

Encontrar un número significativo de defectos durante la prueba de aceptación puede, en algunos casos, considerarse un riesgo importante para el proyecto

La prueba de aceptación también puede satisfacer requisitos o normas legales o reglamentarios.

Las formas comunes de prueba de aceptación incluyen las siguientes:

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

Cada uno de estas se describe en las siguientes cuatro subsecciones

2.2.1 Prueba de Aceptación. Información adicional

Prueba de Aceptación de Usuario (UAT)

- La prueba de aceptación de sistema por parte de los usuarios se centra normalmente en la validación de la idoneidad para el uso del sistema por parte de los usuarios previstos en un entorno operativo real o simulado
- El objetivo principal es crear confianza en que los usuarios pueden utilizar el sistema para satisfacer sus necesidades, cumplir con los requisitos y realizar los procesos de negocio con la mínima dificultad, coste y riesgo

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Aceptación. Información adicional

Prueba de Aceptación Operativa

- La prueba de aceptación de sistema por parte del personal de operaciones o de la administración del sistema se realiza, por lo general, en un entorno de producción (simulado). La prueba se centra en los aspectos operativos:
  - Prueba de copia de seguridad y restauración.
  - Instalación, desinstalación y actualización.
  - Recuperación ante desastres.
  - Gestión de usuarios.
  - Tareas de mantenimiento.
  - Carga de datos y tareas de migración.
  - Comprobación de vulnerabilidades de seguridad.
  - Prueba de rendimiento.
- El objetivo principal de la prueba de aceptación operativa es generar confianza en que los operadores o administradores del sistema pueden mantener el sistema funcionando correctamente para los usuarios en el entorno operativo, incluso en condiciones excepcionales o difíciles

2.2.1 Prueba de Aceptación. Información adicional

Prueba de Aceptación Contractual y Normativa

- La prueba de aceptación contractual:
  - se realiza en función de los criterios de aceptación del contrato para el desarrollo de software a medida
  - los criterios de aceptación deben definirse cuando las partes acuerdan el contrato
  - suele ser realizada por usuarios o por probadores independientes.
- La prueba de aceptación normativa:
  - se lleva a cabo con respecto a cualquier norma que deba cumplirse, como las normas gubernamentales, legales o de seguridad física
  - suele ser realizada por los usuarios o por probadores independientes, en ocasiones los resultados son presenciados o auditados por agencias reguladoras
- El principal objetivo de la prueba de aceptación contractual y normativa es crear confianza en que se ha logrado la conformidad contractual o normativa.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Aceptación. Información adicional

Pruebas Alfa y Beta

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

Pruebas Alfa y Beta

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

II – Prueba a lo largo del CVDS
2.2.1 Prueba de Aceptación. Información adicional

Base de Prueba

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

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

Base de Prueba

Además, como base de prueba para derivar casos de prueba para la prueba de aceptación operativa, se pueden utilizar uno o más de los siguientes productos de trabajo:

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

II – Prueba a lo largo del CVDS
2.2.1 Prueba de Aceptación. Información adicional

Objetos de Prueba Característicos

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

Defectos y Fallos Característicos. Ejemplos

Defectos característicos de cualquier forma de prueba de aceptación:

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

II – Prueba a lo largo del CVDS
2.2.1 Prueba de Aceptación. Información adicional

Enfoques y Responsabilidades Específicos

La prueba de aceptación es a menudo:

- responsabilidad de los clientes, usuarios de negocio, propietarios de producto u operadores de un sistema, y otros implicados
- Considerada como el último nivel de prueba en un ciclo de vida de desarrollo secuencial, pero también pueden ocurrir en otros momentos, por ejemplo:
  - La prueba de aceptación de un producto software comercial de distribución masiva (COTS por sus siglas en inglés) puede tener lugar cuando se instala o integra.
  - La prueba de aceptación de una mejora funcional nueva puede tener lugar antes de la prueba de sistema.

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Aceptación. Información adicional

Enfoques y Responsabilidades Específicos

En el desarrollo iterativo:

- los equipos de proyecto pueden emplear varias formas de prueba de aceptación durante y al final de cada iteración, como las que se centran en verificar una nueva característica en relación con sus criterios de aceptación y las que se centran en validar que una nueva característica satisface las necesidades de los usuarios
- se pueden realizar pruebas alfa y beta, ya sea al final de cada iteración, después de la finalización de cada iteración, o después de una serie de iteraciones
- pueden realizarse la prueba de aceptación de usuario, prueba de aceptación operativa, prueba de aceptación normativa y prueba de aceptación contractual, ya sea al cierre de cada iteración, después de la finalización de cada iteración, o después de una serie de iteraciones

II – Prueba a lo largo del CVDS

2.2.1 Prueba de Aceptación. Información adicional

Información adicional

- En las pruebas de aceptación ¿Se cumplen las expectativas del usuario/cliente antes de su paso a producción ?
- El software personalizado suele ser probado a menudo directamente por el solicitante/cliente
- Para productos "de masas" se involucra a una selección representativa de usuarios en las pruebas
- Las pruebas de aceptación del Contrato se llevan a cabo cuando el software ha sido desarrollado a medida del cliente:
  - Se rigen por los Criterios de Aceptación acordados por contrato.
  - Primeras pruebas en las que el cliente debe estar involucrado activamente
  - Es aconsejable involucrar al cliente en las pruebas durante la fase de desarrollo de prototipos para:
    - Que el cliente determine los casos de prueba para las pruebas de aceptación.
    - Evitar posibles mal interpretaciones de los requisitos – "solo el cliente sabe lo que realmente quiere".
  - Suelen estar cubiertas por el fabricante con sus las pruebas del sistema
  - Los Criterios de Aceptación han de quedar definidos, de forma clara y explícita, en el momento en que ambas partes acuerdan el contrato

Prueba Funcional. Información adicional

- En la prueba funcional de un **sistema**:
  - Se incluye pruebas que evalúan las funciones que el sistema debe realizar
  - Los requisitos funcionales pueden estar descritos en productos de trabajo tales como especificaciones de requisitos de negocio, épicas, historias de usuario, casos de uso, o especificaciones funcionales, o pueden estar sin documentar
  - Las funciones describen "qué" debe hacer el sistema.
- Se deben realizar pruebas funcionales en **todos los niveles de prueba** (p.ej. la prueba de componente puede basarse en una especificación de componentes), aunque el enfoque es diferente en cada nivel
- La prueba funcional **observa el comportamiento del software**, por lo que se pueden utilizar técnicas de caja negra para obtener las condiciones de prueba y los casos de prueba para la funcionalidad del componente o sistema

II – Prueba a lo largo del CVDS
2.2.2 Tipos de Prueba

Prueba Funcional. Información adicional

- Se puede medir la intensidad de la prueba funcional a través de la **cobertura funcional**
  - La cobertura funcional es la medida en que algún tipo de elemento funcional ha sido practicado por pruebas, y se expresa como un porcentaje del tipo o tipos de elemento cubiertos.
  - Por ejemplo, utilizando la trazabilidad entre pruebas y requisitos funcionales, se puede calcular el porcentaje de estos requisitos abordados por las pruebas, identificando potencialmente carencias en la cobertura.
- El diseño y la ejecución de pruebas funcionales pueden implicar competencias o conocimientos especiales, como:
  - el conocimiento del problema de negocio específico que resuelve el software (por ejemplo, el software de modelado geológico para las industrias del petróleo y el gas)
  - el papel particular que desempeña el software (por ejemplo, el software de juegos de azar de ordenador que proporciona entretenimiento interactivo).

Prueba No Funcional. Información adicional

- El diseño y la ejecución de la prueba no funcional pueden implicar competencias o conocimientos especiales, como:
  - el conocimiento de las debilidades inherentes a un diseño o tecnología. Por ejemplo, vulnerabilidades de seguridad asociadas con determinados lenguajes de programación
  - la base de usuarios concreta. Por ejemplo, la "persona" de los usuarios de los sistemas de gestión de centros sanitarios.
- Más información en otros programas de estudio de ISTQB como, por ejemplo, Analista de Pruebas Técnicas

II – Prueba a lo largo del CVDS
2.3.2 Prueba No Funcional

Prueba No Funcional. Información adicional

- El cumplimiento de requisitos no funcionales es igual de importante, pero a menudo difícil de probar y por ello están sometidos a un mayor riesgo.
- En el documento de requisitos no está siempre claro "como de bien" debe funcionar algo:
  - A menudo definiciones vagas (manejar sin problemas, pantallas claras, etc.).
  - Los requisitos no funcionales se dan a menudo de manera implícita y por este motivo no se definen.
- La prueba de un requisito no funcional se da como superada si se consigue un determinado valor en una métrica establecida:
  - P.ej., métrica para la fiabilidad del software:
    - MTBF – Mean Time Between Failure/Tiempo medio entre fallos
    - MTTR – Mean Time To Repair/Tiempo medio de reparación
- De todos modos, suele ser difícil una cuantificación de los requisitos no funcionales.

Características del software. Información adicional

Características **Adecuación Funcional** según ISO/IEC 25010:

- Completitud funcional
- Pertinencia funcional.
  - *¿Son adecuadas las funciones disponibles para la utilización prevista?*
- Corrección funcional.
  - *¿Se ejecutan las funciones correctamente (como estaba acordado)?*

II – Prueba a lo largo del CVDS
Características del software

Características del software. Información adicional

Características no funcionales según ISO/IEC 25010:

- Fiabilidad
  - Capacidad de un software / un sistema de mantener un rendimiento / funcionalidad bajo condiciones predeterminas durante un periodo de tiempo definido.
  - Da una idea del comportamiento de la calidad a lo largo del tiempo.
  - Factores asociados: tolerancia a fallos, capacidad de recuperación ante fallos.
- Usabilidad (ver ISO / IEC 9241)
  - Un software es usable si es:
    - Fácil de entender (uso intuitivo).
    - Fácil de aprender.
  - Existe normativa específica.

II – Prueba a lo largo del CVDS

Características del software

Características del software. Información adicional

Características no funcionales según ISO/IEC 25010:

- Eficiencia de desempeño
  - Utilización de recursos lo más reducida posible (p.ej. tiempo de CPU) para la consecución de una tarea.
- Mantenibilidad
  - Esfuerzo necesario para realizar una serie de modificaciones definidas de antemano.
  - Factores asociados: estabilidad, facilidad de cambio.
- Portabilidad
  - Posibilidad de trasladar un software a otro entorno (hardware, software u organizativo).
  - Factores asociados: facilidad de sustitución, facilidad de instalación, cumplimiento de estándares.
- Seguridad.

¿Están protegidos los datos / programas frente a accesos / pérdidas?

II – Prueba a lo largo del CVDS

Características del software

Características del software. Información adicional

Características no funcionales según ISO/IEC 25010:

- Compatibilidad
  - Interoperabilidad. ¿Se proporciona una interrelación libre de errores con el entorno del sistema?

Consulte el capítulo 4 de ISTQB Analista de Pruebas Técnicas para un mayor detalle de las caracteríticas

II – Prueba a lo largo del CVDS

Características del software

Características del software. Información adicional

- Algunas características de la calidad de software influyen entre sí. Puede ser necesario según el caso asignar unas prioridades a las características.
- Ejemplos:
  - Una descripción puede dejar de ser concisa en aras de la claridad.
  - Un software puede ser redundante (y con ello más difícil de mantener) para aumentar su fiabilidad.
  - Una base de datos lo puede ser para mejorar los tiempos de acceso.
- Para las diferentes características deben ejecutarse diferentes tipos de pruebas.

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba de Caja Blanca. Información adicional

- La prueba de caja blanca obtiene pruebas basadas en la **estructura interna del sistema** o en su implementación.
- La estructura interna puede incluir código, arquitectura, flujos de trabajo y/o flujos de datos dentro del sistema.
- Se puede medir la intensidad de la prueba de caja blanca a través de la **cobertura estructural**.
- La cobertura estructural es la medida en que algún tipo de elemento estructural ha sido practicado mediante pruebas, y se expresa como un porcentaje del tipo de elemento cubierto.
- En el nivel de **prueba de componente:**
  - la cobertura de código se basa en el porcentaje de código del componente que ha sido probado
  - puede medirse en términos de diferentes aspectos del código (elementos de cobertura), tales como el porcentaje de sentencias ejecutables probadas en el componente, o el porcentaje de resultados de decisión probados.
  - estos tipos de cobertura se denominan, de forma colectiva, cobertura de código.

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba de Caja Blanca. Información adicional

- En el nivel de **prueba de integración de componentes** la prueba de caja blanca puede basarse en la arquitectura del sistema, como las interfaces entre componentes y la **cobertura estructural** puede medirse en términos del porcentaje de interfaces practicadas por las pruebas.
- El diseño y la ejecución de la prueba de caja blanca pueden implicar competencias o conocimientos especiales, como:
  - la forma en que se construye el código (por ejemplo, para utilizar herramientas de cobertura de código)
  - cómo se almacenan los datos (por ejemplo, para evaluar posibles consultas a la base de datos)
  - cómo utilizar las herramientas de cobertura e interpretar correctamente sus resultados.

II – Prueba a lo largo del CVDS

2.2.2 Tipos de Prueba

Prueba de Caja Blanca. Información adicional

- Las **pruebas estructurales** (de caja blanca) pueden llevarse a cabo en todos los niveles de prueba.
- Es mejor usarlas después de las técnicas basadas en especificación, ayuda a medir con rigurosidad las pruebas, mediante la evaluación de la cobertura de un tipo de estructura.
- La cobertura mide el grado con el que unas de pruebas ha utilizado una estructura.
- Si la cobertura no es del 100%, pueden diseñarse más pruebas, para comprobar los elementos que se han pasado por alto y, por consiguiente, incrementar la cobertura. Las técnicas de cobertura se tratan posteriormente.
- Las pruebas estructurales pueden basarse en la arquitectura del sistema, como la jerarquía de llamadas.
- Los enfoques de prueba estructural pueden aplicarse también en los niveles de prueba de sistema, integración de sistemas o aceptación (por ejemplo, para los modelos de negocio o las estructuras de menús).

II – Prueba a lo largo del CVDS

2.2.3 Prueba de Confirmación y Prueba de Regresión

Prueba de confirmación. Información adicional

- Repetición de la prueba (Prueba de confirmación)
  - Repetir la ejecución de los casos de prueba que han descubierto errores para comprobar que fueron eliminados.
  - Las repeticiones de las pruebas son, de esta manera, una parte de las **futuras** pruebas de regresión
- Pruebas de la funcionalidad modificada
  - Pruebas de todas las partes del software que han sido modificadas
- Pruebas de nuevas funcionalidades
  - Pruebas completas de las nuevas partes del programa
  - Estas han sido previamente comprobados como componentes etc.
- Pruebas de regresión completas
  - Las pruebas del sistema se ejecutan/repiten de nuevo

II – Prueba a lo largo del CVDS

2.2.3 Prueba de Confirmación y Prueba de Regresión

Prueba de regresión. Información adicional

- Se lleva a cabo cuando se cambia el software o su entorno. La extensión de las pruebas de regresión viene dada por el riesgo de no encontrar defectos en el software que estaba funcionando.
- Las pruebas de regresión pueden hacerse en todos los niveles de prueba y se aplican tanto a pruebas funcionales como a no funcionales y estructurales.
- Los paquetes de pruebas de regresión se lanzan muchas veces y suelen evolucionar lentamente, por lo que son fuertes candidatos a la automatización.
- Nota: Las pruebas de regresión se aplican ya antes de la aceptación del producto

Prueba de regresión. Información adicional

Alcance de las pruebas de regresión:

- La repetición de las pruebas de la parte que generó el error no es suficiente por sí misma para comprobar una modificación.
- Tampoco las pruebas exclusivamente de funciones nuevas o modificadas serían apropiadas en este contexto.
- Unas pruebas completas de regresión serían una posible solución, pero en la práctica necesitan mucho tiempo y generan muchos costes.

Una combinación de pruebas de las partes modificadas (repetición de pruebas de error) y una selección de las pruebas del sistema (p.ej., en función de la priorización de las funciones a comprobar) es lo que mejor se adapta a los requisitos.

Departamento de Formación

www.mtp.es