



- *Indispensable para mejorar las oportunidades laborales en el campo de la Tecnología del Software*
- *Conocimientos indispensables para profesionales vinculados al Aseguramiento de la Calidad*
- *Contiene más de 200 preguntas y respuestas*
- *Incluye dos exámenes de simulación*
- *Ejercicios prácticos para cada tema y un caso real, resuelto, como ejemplo*
- *Más de 150.000 certificados en el mundo ¡No te quedes atrás!*



Rex Black
Gary Rueda Sandoval





Fundamentos de
Pruebas de Software

Rex Black

Presidente de RBCS, Inc.
Ex Presidente de ISTQB
Ex Presidente de ASTQB

Gary Rueda Sandoval

Presidente de Business Innovations S.R.L.
Miembro fundador del HASTQB
Representante del HASTQB Bolivia

Editorial RBCS, Inc.
Texas, Estados Unidos

Rex Black, Gary Rueda Sandoval
FUNDAMENTOS DE PRUEBAS DE SOFTWARE

**No está permitida la reproducción total o parcial de esta obra,
ni su tratamiento o transmisión por cualquier medio o método,
sin autorización escrita de la Editorial.**

DERECHOS RESERVADOS

© 2011 Rex Black, Gary Rueda Sandoval

ISBN: 978-0-9778187-6-1

IMPRESO EN E.E.U.U.

A mi único guía y salvador Jesucristo por mantenerme siempre en el camino de la luz; a mi amada hija Danielita por honrarme con su tierna presencia y gran cariño; a mis padres Nery y Eligia, y a mis hermanos, Adhemar y Yuly, por su siempre incondicional apoyo y por la alegría y los ánimos que me transmiten cada día; y a mi cuñada Sabine, mis amigos y colegas, por su apoyo.

Gary

A Gary Rueda Sandoval por tener la inquietud que condujo a este libro, y por su infinita dedicación en la traducción, la extensión y la adaptación de mi guía Básica de estudio en un libro realmente útil para los profesionales en Pruebas de Software de todo el mundo, que hablan español. A Dena Pauletti y Michelle Egli por sus esfuerzos en hacer este libro una realidad. Finalmente, a mi familia como siempre, especialmente a Laurel, Charlotte y Emma, por su apoyo.

Rex

CONTENIDO

PRÓLOGO

OBJETIVOS
DESTINATARIOS
UNA BREVE HISTORIA DE LAS PRUEBAS DE SOFTWARE

CAPÍTULO 1

FUNDAMENTOS DE PRUEBAS

- 1.1 ¿POR QUÉ SON NECESARIAS LAS PRUEBAS?
 - 1.2 ¿QUÉ SON LAS PRUEBAS?
 - 1.2.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
 - 1.3 PRINCIPIOS GENERALES DE LAS PRUEBAS
 - 1.3.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
 - 1.4 PROCESO DE PRUEBAS BÁSICO
 - 1.4.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
 - 1.5 LA PSICOLOGÍA DE PRUEBAS
 - 1.5.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
 - 1.6 CÓDIGO DE ÉTICAS
- PREGUNTAS DE EXAMEN DE MUESTRA Y SIMULACIÓN
- Sección 1.1: ¿Por qué son necesarias las pruebas? (K2)*
 - Sección 1.2: ¿Qué son las pruebas? (K2)*
 - Sección 1.3: Principios generales de las pruebas (K2)*
 - Sección 1.4: Proceso de pruebas básico (K1)*
 - Sección 1.5: La psicología de las pruebas (K2)*
 - Sección 1.6: Código de éticas (K2)*
 - Capítulo 1 Pregunta que cubre secciones*
 - Preguntas del Examen de simulación 1*
 - Preguntas del Examen de Simulación 2*

CAPÍTULO 2

PRUEBAS A TRAVÉS DEL CICLO DE VIDA DE SOFTWARE

- 2.1 MODELOS DE DESARROLLO DE SOFTWARE
 - 2.1.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
 - Ejercicio 2*
 - Solución del Ejercicio 2*
 - 2.2 NIVELES O FASES DE PRUEBAS
 - 2.2.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
 - 2.3 TIPOS U OBJETIVOS DE LAS PRUEBAS
 - 2.3.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
 - 2.4 PRUEBAS DE MANTENIMIENTO
 - 2.4.1 EJERCICIOS
 - Ejercicio 1*
 - Solución del Ejercicio 1*
- PREGUNTAS DE EXAMEN DE MUESTRA Y SIMULACIÓN
- Sección 2.1 Modelos de desarrollo de software (K2)*
 - Sección 2.2 Niveles de pruebas (K2)*

Sección 2.3: Tipos de pruebas (K2)
Sección 2.4: Pruebas de mantenimiento (K2)
Capítulo 2 Preguntas a través de las secciones
Preguntas del Examen de Simulación 1
Preguntas del Examen de Simulación 2

CAPÍTULO 3

TÉCNICAS ESTÁTICAS

3.1 TÉCNICAS ESTÁTICAS Y EL PROCESO DE PRUEBAS
3.1.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
3.2 PROCESO DE REVISIÓN
3.2.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
3.3 ANÁLISIS ESTÁTICO POR MEDIO DE HERRAMIENTAS
3.3.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
PREGUNTAS DE EXAMEN DE MUESTRA Y SIMULACIÓN
Sección 3.1 Técnicas estáticas y el proceso de pruebas (K2)
Sección 3.2 Proceso de revisión (K2)
Sección 3.3: Análisis estáticos por herramientas (K2)
Capítulo 3 Pregunta a través de las secciones
Preguntas del Examen de Simulación 1
Preguntas del Examen de Simulación 2

CAPÍTULO 4

TÉCNICAS DE DISEÑO DE PRUEBAS

4.1 PROCESO DE DESARROLLO DE PRUEBAS
4.1.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
Ejercicio 2
Solución del Ejercicio 2
4.2 CATEGORÍAS DE LAS TÉCNICAS DE DISEÑO DE PRUEBAS
4.2.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
Ejercicio 2
Solución del Ejercicio 2
Ejercicio 3
Solución del Ejercicio 3
Ejercicio 4
Solución del Ejercicio 4
4.3 TÉCNICAS BASADAS EN LA ESPECIFICACIÓN
4.4 TÉCNICAS BASADAS EN LA ESTRUCTURA
4.4.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
4.5 TÉCNICAS BASADAS EN LA EXPERIENCIA
4.5.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
4.6 SELECCIÓN DE LAS TÉCNICAS DE PRUEBAS
4.6.1 EJERCICIOS
Ejercicio 1
Solución del Ejercicio 1
PREGUNTAS DE EXAMEN DE MUESTRA Y SIMULACIÓN
Sección 4.1 El proceso de desarrollo de pruebas (K3)
Sección 4.2 Categorías de las técnicas de diseño de pruebas (K2)
Sección 4.3 Técnicas basadas en la especificación o de caja negra (K3)
Sección 4.4 Técnicas basadas en la estructura de caja blanca (K3)
Sección 4.5 Técnicas basadas en la experiencia (K2)
Sección 4.6 Selección de las técnicas de pruebas (K2)

Capítulo 4 Pregunta de todas las secciones

Preguntas del Examen de Simulación 1

Preguntas del Examen de Simulación 2

CAPÍTULO 5

GESTIÓN DE PRUEBAS

5.1 ORGANIZACIÓN DE PRUEBAS

5.1.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

5.2 PLANIFICACIÓN Y ESTIMACIÓN DE PRUEBAS

5.2.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

5.3 MONITOREO Y CONTROL DEL PROGRESO DE LAS PRUEBAS

5.3.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

5.4 GESTIÓN DE CONFIGURACIÓN

5.4.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

5.5 RIESGO Y PRUEBAS

5.5.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

5.6 GESTIÓN DE DEFECTOS O INCIDENCIAS

5.6.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

PREGUNTAS DE EXAMEN DE MUESTRA Y SIMULACIÓN

Sección 5.1 Organización de pruebas (K2)

Sección 5.2 Planificación y estimación de pruebas (K2)

Sección 5.3 Monitoreo y control del progreso de pruebas (K2)

Sección 5.4 Gestión de Configuraciones (K2)

Sección 5.5 Riesgos y Pruebas (K2)

Sección 5.6 Gestión de incidencias (K3)

Capítulo 5 Pregunta de todas las secciones

Preguntas del Examen de Simulación 1

Preguntas del Examen de Simulación 2

CAPÍTULO 6

SOPORTE DE HERRAMIENTAS PARA LAS PRUEBAS

6.1 TIPOS DE HERRAMIENTAS DE PRUEBAS

6.1.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

6.2 USO EFECTIVO DE HERRAMIENTAS, LOS BENEFICIOS Y LOS RIEGOS POTENCIALES

6.2.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

6.3 INTRODUCCIÓN DE UNA HERRAMIENTA EN UNA ORGANIZACIÓN

6.3.1 EJERCICIOS

Ejercicio 1

Solución del Ejercicio 1

PREGUNTAS DE EXAMEN DE MUESTRA Y SIMULACIÓN

Sección 6.1 Tipos de herramientas de pruebas (K2)

Sección 6.2: Utilización efectiva de las herramientas: beneficios y riesgos potenciales (K2)

Sección 6.3: Introducción de una herramienta en una organización (K1)

Capítulo 6 Pregunta de todas las secciones

Preguntas del Examen de Simulación 1

Preguntas del Examen de Simulación 2

APÉNDICE A

OMNINET: EL INTERNET EN TODAS PARTES

DOCUMENTO DE LOS REQUISITOS DE MARKETING

1. ALCANCE
 - 1.1 *Términos, Acrónimos y Abreviaciones*
 - 1.2 *Documentos Aplicables*
2. FECHA DE LA VERSIÓN REQUERIDA
3. DESCRIPCIÓN DE LOS REQUISITOS
 - 3.1 *Requisitos técnicos generales*
 - 3.1.1 Bienvenida
 - 3.1.2 Pago
 - 3.1.3 Navegador de Internet
 - 3.1.4 Rendimiento
 - 3.1.5 Localización
 - 3.1.6 Control de Contenido
 - 3.1.7 Terminación de la Sesión
 - 3.1.8 Confidencialidad
 - 3.2 *Administración*
 - 3.2.1 Actualizaciones de Software
 - 3.2.2 Vista de los Quioscos
 - 3.2.3 Vista de los Usuarios
 - 3.2.4 Modificar el Usuario
 - 3.2.5 Terminar el Usuario

APÉNDICE B

OMNINET: EL INTERNET EN TODAS PARTES. DOCUMENTO DE LOS REQUISITOS DEL SISTEMA

- REQUISITOS FUNCIONALES DEL SISTEMA
- REQUISITOS DE FIABILIDAD DEL SISTEMA
- REQUISITOS DE USABILIDAD DEL SISTEMA
- REQUISITOS DE EFICIENCIA DEL SISTEMA
- REQUISITOS DE MANTENIBILIDAD DEL SISTEMA
- REQUISITOS DE PORTABILIDAD DEL SISTEMA
- MODELOS DEL DISEÑO
 - Arquitectura del Sistema Omnitel*
 - Tabla de Decisiones del Procesamiento del Pago*
 - Flujo del Módulo del Quiosco*
 - Diagrama de Transiciones de Estado del Quiosco*
 - Tabla de Transiciones de Estado del Quiosco*
 - Array Ortogonal de la Configuración del Quiosco con respecto al Sistema Operativo del Quiosco/el Navegador/la Velocidad de Conexión*

APÉNDICE C

SOLUCIONES DE LAS PREGUNTAS DE EXAMEN DE MUESTRA Y SIMULACIÓN

- CAPÍTULO 1: FUNDAMENTOS DE PRUEBAS (K2)
 - Sección 1.1: ¿Por qué son las pruebas necesarias? (K2)*
 - Sección 1.2: ¿Qué son las pruebas? (K2)*
 - Sección 1.3: Principios generales de pruebas (K2)*
 - Sección 1.4: Proceso de pruebas básico (K1)*
 - Sección 1.5: La psicología de las pruebas (K2)*
 - Sección 1.6: Código de éticas (K2)*
 - Capítulo 1 Pregunta de todas las secciones*
 - Preguntas del Examen de Simulación 1*
 - Preguntas del Examen de Simulación 2*
- CAPÍTULO 2: PRUEBAS A TRAVÉS DEL CICLO DE VIDA DE SOFTWARE (K2)
 - Sección 2.1 Modelos de desarrollo de software (K2)*
 - Sección 2.2 Niveles de pruebas (K2)*
 - Sección 2.3: Tipos de pruebas: los objetivos de las pruebas (K2)*
 - Sección 2.4: Pruebas de mantenimiento (K2)*
 - Capítulo 2 Pregunta de todas las secciones*
 - Preguntas del Examen de Simulación 1*
 - Preguntas del Examen de Simulación 2*
- CAPÍTULO 3.0: TÉCNICAS ESTÁTICAS (K2)
 - Sección 3.1 Revisiones y el proceso de pruebas (K2)*
 - Sección 3.2 Proceso de revisión (K2)*
 - Sección 3.3: Análisis estático por herramientas (K2)*
 - Capítulo 3 Pregunta de todas las secciones*
 - Preguntas del Examen de Simulación 1*

Preguntas del Examen de Simulación 2

CAPÍTULO 4.0: TÉCNICAS DE DISEÑO DE PRUEBAS (K3)

Sección 4.1 Proceso de desarrollo de pruebas (K3)

Sección 4.2 Categorías de las técnicas de diseño de pruebas (K2)

Sección 4.3 Técnicas basadas en la especificación o de caja negra (K3)

Sección 4.4 Técnicas basadas en la estructura o de caja blanca (K3)

Sección 4.5 Técnicas basadas en la experiencia (K2)

Sección 4.6 Selección de las técnicas de pruebas (K2)

Capítulo 4 Pregunta de todas las secciones

Preguntas del Examen de Simulación 1

Preguntas del Examen de Simulación 2

CAPÍTULO 5.0: GESTIÓN DE PRUEBAS (K3)

Sección 5.1 Organización de las pruebas (K2)

Sección 5.2 Planificación y estimación de pruebas (K2)

Sección 5.3 Monitoreo y control del progreso de las pruebas (K2)

Sección 5.4 Gestión de configuraciones (K2)

Sección 5.5 Riesgo y pruebas (K2)

Sección 5.6 Gestión de incidencias (K3)

Capítulo 5 Pregunta de todas las secciones

Preguntas del Examen de Simulación 1

Preguntas del Examen de Simulación 2

CAPÍTULO 6.0: SOPORTE DE HERRAMIENTAS PARA PRUEBAS (K2)

Sección 6.1 Tipos de herramientas de pruebas (K2)

Sección 6.2: Utilización efectiva de herramientas: beneficios y riesgos potenciales (K2)

Sección 6.3: Introducción de una herramienta en una organización (K1)

Capítulo 6 Pregunta de todas las secciones

Preguntas del Examen de Simulación 1

Preguntas del Examen de Simulación 2

APÉNDICE D

LISTA DE LOS ESTÁNDARES DE PRUEBAS

APÉNDICE E

PREPARACIÓN PARA EL EXAMEN

APÉNDICE F

ACRÓNIMOS

APÉNDICE G

BIBLIOGRAFÍA

PERFIL DE RBCS

PERFIL DE BUSINESS INNOVATIONS

ÍNDICE ANALÍTICO

Acerca de los Autores

Prólogo

En los últimos años hemos observado que la producción de software ha crecido exponencialmente en todo el mundo. La cantidad de software en todo tipo de dispositivos está aumentando rápidamente. Por ejemplo, se espera que la cantidad de código fuente en un teléfono móvil aumente a 10 millones de líneas de código entre los años 2010 y 2011¹. En el mismo período, el software de un automóvil podría crecer hasta un máximo de 100 millones de líneas de código. El sistema de software del control de vuelo del Boeing 787 consta de 6,5 millones de líneas de código, cerca de tres veces más que el Boeing 777². Todo tipo de nuevas aplicaciones e infraestructuras dependen en gran medida del software, incluidas las redes sociales, el video por Internet, los sistemas de alerta temprana, los sistemas de información médica y los sistemas financieros. A raíz de este crecimiento las exigencias de la calidad del software ahora son mayores y el cliente espera el menor número de fallas posibles cuando el software ya está funcionando con clientes finales, por lo tanto eso significa mayores ganancias en el negocio y la reducción de los riesgos de las pérdidas económicas, de tiempo y reputación, los desastres ambientales, las lesiones o la muerte. Por ejemplo:

“Millones de cuentas bancarias fueron impactadas por defectos debido a la instalación de un código de software probado incorrectamente en el sistema de procesamiento de transacciones del banco *North American Bank* en E.E.U.U. Las pérdidas excedieron los 100 millones de dólares”.

No hay suficientes estadísticas publicadas acerca del porcentaje de instituciones que han tenido problemas en el software pero eso no quiere decir que no hay defectos o pérdidas en el negocio.

Esos incidentes originan una serie de preguntas. ¿Por qué los problemas del software no fueron detectados antes a pesar de todas las herramientas modernas a disposición de los ingenieros de sistemas? ¿Cómo puede ser remediada esta situación? Una gran parte de las respuestas a estas preguntas caen en una de las áreas más desatendidas del desarrollo y despliegue de productos de software—*Las Pruebas de Software*.

Las Pruebas de Software son predominantemente consideradas como una actividad periférica, casi una formalidad, antes del despliegue del software. Un cambio de actitud hacia las Pruebas de Software puede reducir tremadamente los problemas normalmente asociados con el lanzamiento del nuevo software y minimizar los riesgos implicados. Las Pruebas de Software consisten en un proceso crítico para asegurar que el software sea entregado al cliente libre de defectos, y debería ser tratado como tal.

En la actualidad la mayoría de las organizaciones no cuentan con un personal profesional en Pruebas de Software para gestionar un proceso de control de calidad paralelo al proceso de desarrollo porque las empresas tercerizadas están más orientadas al desarrollo por lo cual hay una cierta parcialización hacia el software desarrollado por parte del fabricante. Las Pruebas de Software requieren un conocimiento especializado y profesional con una metodología diferente que escribir código o construir software. Un peligro inherente de contratar los servicios de control de calidad del software del mismo fabricante es el conflicto de interés.

Por lo tanto, debido a este conflicto de interés y la criticidad del software tiene sentido la separación del desarrollo y las Pruebas de Software en equipos separados y además las Pruebas de Software deben ser llevadas a cabo por personas que conocen las técnicas y métodos, y tienen experiencia y conocimiento del dominio del negocio y las herramientas de Pruebas de Software. Entonces en el mundo se han creado diversos programas estándar de estudios para satisfacer las necesidades en la calidad del software, de los cuales uno de los más reconocidos a nivel mundial es la Certificación en Pruebas de Software (“Software Testing Certification”) del ISTQB (“International Software Testing Qualifications Board”) que propone diversos cursos de estudios para distintos niveles de experticia, así como el nivel básico, tres niveles avanzados, el nivel avanzado completo y el nivel experto. En este libro trataremos el nivel básico según el programa de estudios 2011.

Por Gary Rueda Sandoval

Objetivos

Este libro proporciona a los Ingenieros de Pruebas y Jefes de Pruebas el fundamento, los procesos, las herramientas y habilidades esenciales que ellos necesitan para posicionarse en un camino hacia el verdadero profesionalismo en pruebas. Este libro práctico cubre las principales técnicas de diseño de pruebas por medio de la clase y los ejercicios. El libro proporciona la metodología detrás de un programa de pruebas exitoso y cubre una amplia gama de temas, desde aquellos relacionados con el probador individual hasta los relacionados con el departamento de pruebas en su totalidad. El proceso de pruebas es presentado, tanto a través de la teoría como mediante ejercicios prácticos que siguen un proyecto como ejemplo, incluyendo las tareas difíciles como el seguimiento y la presentación de los resultados de las pruebas. Así mismo incluye la creación de un entorno de pruebas y automatización de pruebas junto con los ciclos de vida del desarrollo de

sistemas y cómo estos afectan a las pruebas.

Este libro le proporciona una guía de auto-estudio para pasar el examen Probador Certificado ISTQB Nivel Básico (ISTQB Certified Tester Foundation Level") que incluye los seis capítulos, un glosario de los términos más importantes, ejercicios prácticos y sus soluciones por cada capítulo junto con dos exámenes de simulación y con más de 200 preguntas de muestra que abarcan cada objetivo del aprendizaje del Programa de Estudios ("Syllabus")³, una orientación acerca de cómo prepararse para el examen, y más.

A través de los capítulos, la argumentación y los ejercicios prácticos, usted aprenderá a:

- Explicar los efectos y el daño que los defectos pueden causar.
- Articular la necesidad de las pruebas.
- Describir el rol de las pruebas en el aseguramiento de la calidad.
- Identificar los objetivos, los principios y los propósitos comunes de las pruebas.
- Introducir procesos de pruebas estructurados, pre-planificados.
- Adaptarse a los factores psicológicos y gestionarlos para el éxito de las pruebas.
- Relacionar las actividades de desarrollo y pruebas.
- Adaptar los modelos de desarrollo de software al contexto del proyecto y el producto.
- Seleccionar e implementar niveles o fases adecuadas de las pruebas, con los participantes, los objetivos, las metas y los ítems apropiados sometido a prueba para cada nivel o fase de prueba.
- Seleccionar y planificar los principales tipos u objetivos de las pruebas, incluyendo las pruebas funcionales y no funcionales, las pruebas estructurales, las pruebas de confirmación y las pruebas de regresión.
- Explicar las razones para las pruebas de mantenimiento y cómo las pruebas de mantenimiento se diferencian de las pruebas de aplicaciones nuevas.
- Entender el valor, la importancia y la utilización de las técnicas estáticas y el análisis estático, y la diferencia entre las técnicas estáticas y dinámicas.
- Explicar las fases, los roles y las responsabilidades de una revisión formal típica, y comparar los diferentes tipos de revisiones.
- Comprender los factores para las revisiones exitosas.
- Comprender y realizar un análisis de los riesgos de calidad para que sirva como la base para las pruebas, utilizando los factores de probabilidad e impacto para determinar el nivel de riesgo.
- Escribir diseños, casos y procedimientos de prueba, relacionarlos entre sí, y rastrear estos ítems a la base de las pruebas.
- Desarrollar un cronograma de ejecución de las pruebas.
- Explicar las características, las diferencias y las razones para las pruebas basadas en la especificación (caja negra), basadas en la estructura (caja blanca), y basadas en la experiencia.
- Escribir casos de prueba utilizando el particionamiento⁴ ("partitioning") de equivalencias, el análisis de valores límite, las tablas de decisión y los diagramas de transición de estados, comprendiendo el propósito principal de cada técnica y qué cantidad de cobertura es suficiente para cada técnica.
- Escribir y medir los casos de prueba utilizando los conceptos de pruebas estructurales como la cobertura, la cobertura de sentencia y decisión y otras técnicas de diseño de pruebas del flujo de control.
- Comprender los factores que influyen en la selección de las técnicas apropiadas de diseño de pruebas.
- Explicar la importancia de las pruebas independientes.
- Comprender los beneficios y las desventajas de las pruebas independientes.
- Seleccionar los distintos miembros del equipo para su inclusión en un equipo de pruebas.
- Saber las tareas de un líder y probador típico.
- Comprender y escribir varios tipos de planes de pruebas dependiendo del proyecto, los niveles y los objetivos.
- Estimación de las pruebas a través de las métricas y la experticia, y reconocer los factores que afectan una estimación.
- Comprender, utilizar e interpretar las métricas comunes para monitorear la preparación y la ejecución de las pruebas.
- Explicar cómo la gestión de configuración apoya a las pruebas.
- Saber los daños típicos y riesgos potenciales para las pruebas.
- Diferenciar entre los riesgos de proyecto y calidad (del producto).
- Escribir un buen informe de defecto o incidencia, con el contenido apropiado.
- Conocer los diferentes tipos de las herramientas de pruebas, incluyendo las herramientas de pruebas de los programadores.
- Explicar las diferentes técnicas de guiones para las herramientas de ejecución de pruebas, incluyendo las dirigidas por datos y las dirigidas por palabras clave.
- Saber los beneficios y los riesgos potenciales de la automatización de pruebas.
- Planificar la introducción de una herramienta de pruebas en una organización.

- Formular las metas de una prueba de concepto para la evaluación de una herramienta de pruebas.
- Explicar los factores necesarios para un buen soporte de herramientas.

Destinatarios

Este libro no es sólo para los probadores sino también para quienes están encargados de la adquisición de software en general, gerentes de tecnología, gerentes del Aseguramiento de la Calidad/Control de la Calidad ("QA/QC"), gerentes de sistemas, jefes de proyectos de software, analistas, arquitectos, desarrolladores, estudiantes y profesores de TI.

Una breve historia de las Pruebas de Software

Desde los comienzos de la computación hubo pruebas. La siguiente clasificación (por supuesto hay más) fue creada en 1988 por D. Gelperin y W.C. Hetzel⁵. Ellos clasificaron las pruebas de software con las siguientes fases y metas:

Hasta 1956 fue el período orientado a la depuración, donde las pruebas eran asociadas a la depuración: no había una clara diferencia entre las pruebas y la depuración.

De 1957 a 1978 fue el período orientado a la demostración, donde la depuración y las pruebas fueron diferenciados ahora - en este período fue demostrado que el software satisface los requisitos.

De 1979 a 1982 fue el período orientado a la destrucción, donde la meta era de encontrar errores.

De 1983 a 1987 fue el período orientado a la evaluación: la intención aquí es de que durante el ciclo de vida del software una evaluación del producto es proveído y una medición de su calidad.

De 1988 hasta la fecha fue el período de la prevención donde las pruebas estarían para demostrar que el software satisface su especificación, para detectar y prevenir los defectos.

¹ 1. R.N. Charette, "Why Software Fails," IEEE Spectrum, vol. 42, no. 9, 2005, pp. 42-49.

² M. Mecham, "Boeing Faces 'Pretty Tight' 787 Delivery Schedule," Aviation Week, 9 Sept. 2007

³ Es un programa, esquema o plan de estudios de un texto, una conferencia o curso de estudio.

⁴ Aunque no está definido en la Real Academia Española, es un término muy utilizado para indicar el sustantivo de la acción de partir algo en particiones.

⁵ Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". CACM 31 (6). ISSN 0001-0782.

Capítulo 1

Fundamentos de Pruebas

"La primera demostración en el espacio que vamos a hacer nosotros es en el año 2014 en la Estación Espacial Internacional (ISS). Vamos a montar un pequeño prototipo del motor, para probarlo y dispararlo en la ISS, y verificar el rendimiento, su fiabilidad y capacidad de operar tal como estamos prediciendo"

Franklin Chang Díaz, primer astronauta latinoamericano de la NASA e inventor del motor de cohete VASIMR.

El capítulo 1, Fundamentos de Pruebas, contiene las siguientes seis secciones:

1. ¿Por qué son necesarias las pruebas?
2. ¿Qué son las pruebas?
3. Principios generales de las pruebas.
4. Proceso de pruebas básico.
5. Psicología de las pruebas.
6. Código de éticas.

La mayoría de las secciones estarán desglosadas en dos o más partes.

1.1 ¿Por qué son Necesarias las Pruebas?

Objetivos del Aprendizaje

LO-1.1.1 Describir con ejemplos, la manera en la cual un defecto en el software puede provocar daño a una persona, al entorno o una compañía. (K2) 6

LO-1.1.2 Distinguir entre la causa raíz de un defecto y sus efectos. (K2)

LO-1.1.3 Dar razones por qué las pruebas son necesarias proporcionando ejemplos. (K2)

LO-1.1.4 Describir por qué las pruebas son parte del aseguramiento de la calidad y dar ejemplos de cómo las pruebas contribuyen a la más alta calidad. (K2)

LO-1.1.5 Explicar y comparar los términos error, defecto, falta, falla y los términos correspondientes equivocación y "bug" utilizando ejemplos. (K2)

Esta sección, ¿Por qué son necesarias las pruebas?, abordará los siguientes conceptos:

- ¿Cómo pueden los defectos causar daño o perjuicio?
- Los defectos y sus efectos.
- La necesidad de las pruebas.
- El rol de las pruebas en el aseguramiento de la calidad.

Glosario del ISTQB

"Bug": Véase *defecto*.

Defecto: Un desperfecto que puede causar que el componente o el sistema falle al realizar su función requerida, p.ej. una sentencia incorrecta o una definición incorrecta de los datos. Si es encontrado un defecto durante la ejecución, puede causar una falla del componente o sistema.

Error: Una acción humana que produce un resultado incorrecto.

Falla: Desviación del componente o el sistema de su prevista entrega, servicio o resultado.

Falta: Véase *defecto*.

Equivocación: Véase *error*. Glosario del ISTQB

Calidad: El grado en el cual un componente, un sistema o un proceso cumple los requisitos especificados y/o las necesidades y las expectativas del usuario/cliente.

Requisito: Una condición o una capacidad necesitada por un usuario para resolver un problema o lograr un objetivo que debe ser cumplido o que debe poseer un sistema o un componente de sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto.

El software está en todo lugar. Se podría decir que está presente en todos los aspectos y momentos de la vida de las personas o mejor dicho en casi todos. Algunas veces el software es obvio, otras veces no. Cuando estamos en un banco o estamos utilizando un cajero automático, podemos observar el funcionamiento de los sistemas de software. Pero cuando estamos manejando un auto moderno, es fácil de olvidar que hay más poder computacional y presencia de software en la mayoría de los autos que en la mayoría de las naves espaciales construidas hasta la década de los ochenta.

Si bien es fácil de olvidar el software cuando funciona, también es fácil para la mayoría de nosotros de recordar nuestras experiencias acerca del software que no funcionó. Una serie de dificultades pueden suceder a varias personas cuando un software contiene defectos. Empecemos con la compañía u organización que desarrolla o adquiere el software.

Esa compañía puede sufrir daños acerca de su reputación ya sea si es un fabricante o una compradora de software cuyos clientes sufren.

La compañía puede sufrir altos o impredecibles costos de mantenimiento cuando las fallas ocurren en la producción.

Pueden ocurrir retrasos inesperados en los ciclos de versiones cuando los problemas son descubiertos tarde en el ciclo de desarrollo.

Si un gran número de defectos aparecen antes de la versión o después de la versión, esto puede conducir a una falta de confianza por parte de los interesados del negocio ya sea si ellos son empleados de la compañía o clientes de la misma.

Por supuesto, que en ciertas situaciones el software con muchos defectos puede resultar en juicios legales.

Por otro lado, en cuanto al entorno, el software está más y más embebido en sistemas que controlan procesos industriales, fábricas, plantas de energía y por supuesto automóviles. Cuando estas aplicaciones no funcionan correctamente, eso puede resultar en un exceso de contaminación y desperdicio de recursos. Por ejemplo, un sistema podría quemar más gasolina o consumir más electricidad de lo necesario.

Para los individuos, las sociedades y los estados existen las amenazas de las fallas que son causadas por los defectos. Las personas pueden perder sus trabajos cuando las fallas en un software se convierten en un revés económico para su empleador.

En algunos casos las personas pueden perder sus vidas debido al software. Tuvimos recientemente un incidente en un hospital de Texas, cuando una persona al intentar subir al elevador mientras las puertas se cerraban, ésta quedó con la mitad de su cuerpo dentro. El software que controlaba al elevador no evitó correctamente el movimiento con la puerta parcialmente abierta.

Así mismo la gente puede perder sus derechos civiles como ha pasado cuando las máquinas de las elecciones computarizadas fracasaron.

Muchas personas han sufrido la pérdida de la privacidad, el robo de la identidad y así sucesivamente debido a que el software utilizado por sus bancos y universidades tenía defectos. Misiones o incluso guerras se podrían perder debido a los defectos a medida que el software se ha hecho más frecuente en los entornos de las zonas de guerra. Con frecuencia, incluso los soldados individuales tienen software ejecutándose en sistemas que ellos transportan o de los que ellos dependen.

¿De dónde vienen todos estos defectos y qué es lo que hacen?

Los defectos no son introducidos por pequeñas hadas en los sistemas que vuelan alrededor de los centros de datos y lanzan los defectos en las computadoras y el código. Para ser completamente franco al respecto, los defectos están en el sistema porque alguien los colocó allí. Alguien cometió una equivocación, un error, cuyo resultado fue la inserción de un defecto o "bug" en el sistema.

Un defecto o un error pueden ser insertados en cualquier momento durante el ciclo de vida. Pueden ser insertados en los requisitos o en las especificaciones de diseño. Pueden ser insertados en el código, ya sea en la lógica de negocios o en la interfaz de usuario. Pueden ser insertados en la documentación, ya sea en la documentación electrónica o impresa. Ahora, una vez que el defecto es insertado en el sistema, éste podría o no podría de verdad resultar en una falla. Para que una falla ocurra, la parte defectuosa del sistema tiene que ser ejecutada. En ese momento el defecto podría provocar que el sistema funcione incorrectamente. Asumiendo que las precondiciones correctas están vigentes, el sistema podría fallar al no cumplir con lo que debería hacer en esas condiciones. Si esa falla es visible o llega a ser visible después, ya sea a un cliente, un usuario u otro interesado del negocio, ese comportamiento podría resultar en la insatisfacción de la calidad del sistema.

Note que hay mucho uso del verbo condicional "podría". Bajo muchas circunstancias un defecto podría existir en el sistema y todavía no ser visible en cuanto al comportamiento visible, simplemente porque requiere una secuencia específica de acciones antes de la ejecución del código que contiene el defecto, o los datos específicos que son proporcionados al código que contiene el defecto o ambos.

Entonces, ¿Qué hay detrás de todos estos defectos y fallas?

Los defectos ocurren por varias razones.

Los programadores, los analistas de negocios, los analistas de sistemas y otros colaboradores individuales, incluyendo a los probadores, son falibles. Cometen errores y algunos de esos errores introducen los defectos.

La gente tiende a cometer más errores cuando está bajo condiciones de presión de tiempo. Hoy en día las condiciones de presión de tiempo están omnipresentes en los proyectos de desarrollo de software y de sistemas.

Es más fácil cometer un error cuando se está tratando de resolver un problema complejo, o cuando se está utilizando un código que ha llegado a ser complejo a través de los años, o cuando se está utilizando una infraestructura que es complicada.

En muchos casos, muchas partes diferentes del sistema o de los sistemas de sistemas tienen que trabajar todas juntas. Los sistemas de sistemas y las tecnologías que los implementan están cambiando continuamente y deben engranar. Por ejemplo, considere lo que se está haciendo estos días con las arquitecturas orientadas al servicio. A menudo existen intentos de hacer disponible los

servicios implementados en código de mainframes, frecuentemente código COBOL antiguo que se ejecuta en mainframes, a veces código que ha estado en los centros de datos por 20, 30 o más años.

Además, tenemos muchos sistemas que interactúan de punta a punta, donde una transacción tiene que ser manipulada por una serie de sistemas antes que la transacción se complete correctamente. Cada operación podría ser simple, pero la secuencia total de operaciones puede llegar a ser compleja, y esa complejidad puede conducir a expectativas equivocadas por parte de una o más personas. Esto conduce a un defecto en la interacción o la secuenciación de las operaciones.

Ahora, muchas fallas ocurren debido a los defectos como fueron descritos en el párrafo anterior. Sin embargo, estos también ocurren debido a las condiciones del entorno. Por ejemplo, el calor excesivo en el sistema que ejecuta el código puede provocar fallas intermitentes en la memoria, el almacenamiento o el CPU. Es decir, el uso incorrecto del software, si es intencional o accidental, contiene el defecto, o ambos.

Uno podría argumentar que tal vez el sistema debería controlar todas las circunstancias y los usos previsibles. Sin embargo, dado el conjunto casi infinito de condiciones y entradas con los que el sistema podría estar confrontado, no es completamente realista esperar que el uso incorrecto no provoque fallas en algunas circunstancias.

Entonces habiendo considerado los defectos y las fallas, podemos ver que el sistema está sujeto a varios riesgos.

Glosario del ISTQB

Riesgo: Un factor que podría resultar en futuras consecuencias negativas; usualmente expresado como el impacto y la probabilidad.

Los riesgos acerca de los cuales hemos estado hablando hasta ahora se relacionan con la calidad. Estamos utilizando la calidad en el sentido amplio de: "¿Está listo el sistema para el cliente, la versión o el siguiente paso en el proceso?" Seremos más precisos en la utilización de esta palabra importante, calidad, más adelante.

Existen otros riesgos y otras restricciones en los proyectos de software que deben ser considerados. Por ejemplo, existen riesgos relacionados con el fracaso de implementar todas las características necesarias. Existen riesgos relacionados con versiones atrasadas. Existen riesgos relacionados con el presupuesto y con las restricciones de los recursos. Las pruebas no gestionan mucho estas áreas de riesgos, sino más bien son afectadas por estos riesgos. Hablaremos después más de este tema.

Sin embargo piense por el momento acerca de las pruebas como un medio de gestionar los riesgos de calidad. Las pruebas hacen esto de varias formas. Una forma es de proporcionar la información para guiar el proyecto en el área de los riesgos de calidad. Otra es de enfocarse en las pruebas acerca de los riesgos de calidad más importantes.

Aún otra es de localizar y dejar a las personas que reparen los problemas importantes.

Las pruebas pueden y algunas veces también deben ser realizadas debido a las regulaciones, las cuestiones de conformidad, o los contratos. Por ejemplo, las regulaciones Sarbanes-Oxley son aplicadas en muchas instituciones financieras en los Estados Unidos. Para ciertos tipos de sitios web, la accesibilidad para personas con discapacidades podría ser necesaria.

Está bien, regresemos a esta pregunta ¿Qué es la calidad? Existen dos definiciones formales comunes para la calidad las cuales se diferencian claramente.

Una es "la aptitud para el uso". La otra es "la conformidad con los requisitos".

Decimos que éstas son claramente diferentes porque la definición "apta para el uso" implica que aquellos que usan, emplean, adquieren o necesitan el software son aquellos que deberían determinar si tiene o no calidad. La definición "Conformidad con los requisitos", por el contrario, necesita simplemente que el sistema trabaje de la manera especificada en algún documento. Esto deja abierta la pregunta de que si la definición de la capacidad de ser correcto ("correctness") que es proporcionada en el documento es correcta por sí misma.

¿Cómo se relacionan las pruebas con la calidad? En un párrafo anterior, mencionamos que las pruebas son como un medio para gestionar los riesgos de calidad. Vayamos un poco más a fondo.

Suponga que ejecutamos un conjunto de pruebas las cuales encuentran muy pocos defectos. En tal caso, deberíamos tener un nivel más alto de confianza en el sistema.

Suponga que ejecutamos un conjunto de pruebas y todas esas pruebas pasan, en este caso el nivel restante del riesgo de la calidad es bajo, con la condición de que las pruebas fueron diseñadas correctamente.

Suponga que ejecutamos un conjunto de pruebas y esas pruebas fallan. Esas pruebas nos han proporcionado ahora la oportunidad de mejorar la calidad del sistema. Finalmente, si su cobertura es adecuada entonces la totalidad del conjunto de pruebas nos dan una buena evaluación de la calidad del sistema.

Por supuesto, esto nos lleva a la pregunta, "¿Estamos realmente probando las cosas correctas?"

Piense que las características de calidad son importantes para su sistema y por consecuencia resultará un sistema el cuál es apto para el uso. Las características de calidad como el rendimiento, la usabilidad, la fiabilidad y la exactitud. ¿Ha pensado en lo que son todas esas características? ¿Está actualmente probándolas lo suficiente? ¿Cómo lo sabe?

Más adelante en este libro, veremos las formas de determinar las respuestas a esas preguntas.

Es importante ser claro en la relación y diferencia entre las pruebas, el aseguramiento de la calidad y el mejoramiento de la calidad. Los grupos de pruebas son con frecuencia referidos erróneamente como grupos de "aseguramiento de la calidad" o grupos de "QA". Mientras que las pruebas pueden y deberían ser parte de una estrategia más grande del aseguramiento de la calidad, las pruebas no aseguran la calidad más que la subida en una báscula cada mañana nos asegura perder peso. Sin embargo sólo con la atención diligente a la calidad en todo el ciclo de vida del software y en toda la organización podemos asegurar la calidad.

El aseguramiento de la calidad por sí misma tampoco es suficiente. A largo plazo, queremos la calidad para mejorar. Para hacer eso, necesitamos examinar esas áreas donde necesitamos mejorar y encontrar la forma de realizar las mejoras adecuadas. Las pruebas proporcionan importantes entradas para el mejoramiento de la calidad. Por ejemplo cuando encontramos muchos defectos, si los clasificamos adecuadamente, y los programadores los clasifican adecuadamente, se pueden tomar pasos para reducir el número de tales defectos en el futuro.

1.2 ¿Qué son las Pruebas?

Objetivos del Aprendizaje

LO-1.2.1 Recordar los objetivos comunes de las pruebas. (K1)

LO-1.2.2 Proporcionar ejemplos para los objetivos de las pruebas en las diferentes fases del ciclo de vida del software. (K2)

LO-1.2.3 Diferenciar las pruebas de la depuración. (K2)

Glosario del ISTQB

Depuración: El proceso de encontrar, analizar y retirar las causas de las fallas en el software.

Esta sección, ¿Qué son las Pruebas?, abordará los siguientes conceptos clave:

- ¿Cuáles son los objetivos comunes de las pruebas?
- ¿Cuál es el propósito de las pruebas en el desarrollo de software, el mantenimiento y las actividades de operaciones?
- ¿Cuál es el propósito de las pruebas en cuanto a la localización de defectos, el fomento de la confianza, la entrega de información y la prevención de defectos?

En muchos casos, las pruebas tienen uno o más de los siguientes objetivos generales:

Las pruebas podrían ser llevadas a cabo para encontrar defectos y luego proporcionar a los programadores la información que ellos necesitan para corregir estos defectos. Los programadores no resuelven todos los defectos usualmente, pero nuestra información debería ayudarles a corregir al menos los defectos más importantes.

También es cierto, que las pruebas podrían ser llevadas a cabo para dar a la gente confianza en el nivel de calidad del sistema. Cuando las compañías adquieren aplicaciones, éstas ejecutan a menudo pruebas de aceptación para ganar la confianza previa al despliegue de la aplicación en el centro de datos.

Así mismo las pruebas podrían ser llevadas a cabo para prevenir defectos. Esto le podría parecer extraño a usted. Sin embargo, las pruebas previenen defectos cuando los probadores están involucrados en las revisiones y cuando los diseños de las pruebas son completados en paralelo con la implementación del sistema. Estas actividades de pruebas tempranas crean ciclos de retroalimentación beneficiosos entre las actividades de pruebas y las actividades de desarrollo, limpiando los defectos temprano en el ciclo de vida.

Finalmente las pruebas podrían ser llevadas a cabo para proporcionar información, obtener una visión acerca de lo que realmente importa a la calidad del sistema sometido a pruebas. ¿Estamos listos para enviar? ¿Cuándo estaremos listos para enviar? ¿Cuáles son las áreas de riesgo que todavía tenemos que abordar? ¿Cuáles son los problemas conocidos más temidos?

Una vez que empecemos a generar información como esa, la información que conceda una visión del nivel de calidad, podríamos ir tras el siguiente objetivo, el de ayudar a la gerencia a comprender la calidad en el contexto de los objetivos más grandes del proyecto. Sólo cuando la gerencia comprenda nuestros hallazgos y lo que estos significan en cuanto al proyecto, podemos decir que como probadores estamos ayudando verdaderamente a gestionar los riesgos de calidad.

Esta lista de objetivos no es en absoluto exhaustiva, usted podría ser capaz de pensar en otros objetivos que tuvo acerca de sus propios proyectos. Si es así, está bien.

Lo importante no es tanto los objetivos de pruebas que son escogidos, sino el alineamiento de esos

objetivos. Muchas cosas pueden salir mal aquí. Usted podría desalinear sus propios planes y acciones como un profesional de pruebas con los objetivos de las pruebas que son establecidos por la gerencia. Usted mismo podría verse forzado a tratar de alinear sus planes y acciones con los objetivos de pruebas los cuales están sólo implícitos, no declarados por la gerencia, conduciendo a problemas cuando usted se equivoque. Peor aún, es posible que no haya un sólo conjunto de objetivos de pruebas consistente a través de toda la gerencia, sino más bien un conjunto de objetivos de pruebas diferente y posiblemente contradictorio a través de los diferentes jefes y otros accionistas.

Estos problemas de alineación son comunes y son problemas muy dañinos para los grupos de pruebas.

Los objetivos de las pruebas pueden cambiar dependiendo de la fase o del nivel de pruebas con el que estemos involucrados. Por ejemplo, considere el objetivo de las pruebas de encontrar defectos o "bugs".

Glosario del ISTQB

Objetivo de prueba: Una razón o propósito para el diseño y la ejecución de una prueba.

Para las pruebas unitarias o de componente, usted podría encontrar defectos en las partes individuales del sistema sometidos a pruebas antes de que las partes estén totalmente integradas al mismo.

Para las pruebas de integración o cadena, es posible que quiera encontrar defectos en las relaciones e interfaces entre los pares y grupos de componentes en el sistema sometido a pruebas a medida que las partes se juntan.

Para las pruebas de sistema, es posible que quiera encontrar defectos en los comportamientos, funciones y respuestas generales y particulares del sistema sometido a pruebas en su totalidad.

Para las pruebas piloto o de aceptación, por lo general no deseamos encontrar defectos en absoluto. Usualmente es un problema si encontramos defectos. Más bien, queremos demostrar que el producto está listo para su despliegue o versión, o evaluar la calidad y dar información acerca del riesgo del despliegue o la versión.

Para las pruebas de mantenimiento, podríamos estar buscando especialmente un tipo particular de defecto, aquellos introducidos durante el desarrollo de los cambios.

Finalmente, para las pruebas operativas, queremos buscar otra vez un tipo particular de defectos, especialmente aquellos relacionados con las características no funcionales del sistema como la fiabilidad o la disponibilidad, usualmente en el entorno en funcionamiento.

Como puede ver, un objetivo general debe ser adaptado a un objetivo específico basado en el contexto en el cual es aplicado.

Por supuesto, queremos ser probadores de software, efectivos y eficientes, pero muchas veces somos descuidados cuando usamos estas palabras, "efectivo" y "eficiente". Veamos si podemos definirlas estrictamente.

Por efectivo, queremos decir producir un resultado decidido, decisivo o esperado; notable. Entonces para ser probadores efectivos, debemos seleccionar los objetivos correctos y los resultados esperados. Para que todos concuerden con que somos efectivos, cada uno tendría que estar de acuerdo con los objetivos.

Por eficiente, queremos decir productivo del efecto esperado, especialmente productivo sin pérdida. Entonces para ser probadores eficientes, debemos asignar los recursos adecuadamente. Los recursos incluirían el tiempo, ambos en cuanto al esfuerzo y la duración, así como también al dinero y aquellas cosas que cuestan dinero.

Ahora, es importante tener en cuenta que la optimización de la efectividad o eficiencia de un conjunto de actividades podría ser nocivo en realidad para la efectividad y eficiencia de un trabajo más grande. Por ejemplo, suponga que decidimos no empezar con el diseño de pruebas hasta que la especificación de requisitos esté terminada. Eso podría mejorar nuestra propia eficiencia, porque no tenemos que rehacer nuestras pruebas después cuando la especificación cambie. Sin embargo, lo que perdemos es la oportunidad de encontrar defectos en la especificación de requisitos mientras está siendo escrita. Perdemos la oportunidad de prevenir defectos.

Éste es un ejemplo clásico de suboptimización. Otro ejemplo clásico es cuando los programadores dicen "¡Oigan!, no necesito realizar las pruebas unitarias, porque los probadores encontrarán los defectos después". Eso es verdad, pero los probadores no necesariamente encontrarán todos los defectos que pudieran haber sido encontrados por medio de las pruebas unitarias, y además los probadores encontrarán los defectos a un costo más alto.

Entonces, piense en efectividad y eficiencia en el contexto del proceso completo de desarrollo o mantenimiento, no sólo en el de su proceso de pruebas.

Es muy importante distinguir entre las pruebas y la depuración.

Las pruebas, como hemos visto, tienen una serie de objetivos. Uno de esos objetivos es encontrar

defectos. Por supuesto, no encontramos realmente defectos al probar por lo general. Encontramos fallas que han ocurrido, a menudo debido a defectos.

La depuración es una actividad específica, activada por el descubrimiento de una falla que creemos que surge de un defecto. Implica la identificación de la causa raíz de un defecto. Luego, la causa raíz es eliminada y el código reparado. Finalmente, la depuración, realizada correctamente, incluye algún nivel de pruebas unitarias para comprobar si la reparación fue correcta.

Una vez que la corrección de un defecto es devuelta al probador quién lo descubrió, ese probador realiza luego la prueba de confirmación para asegurar que la falla observada previamente haya sido resuelta. Esta prueba de confirmación es también referida como la repetición de una prueba en el glosario del ISTQB.

Note que en este proceso hay diferentes responsabilidades. Los probadores prueban, mientras que los programadores depuran.

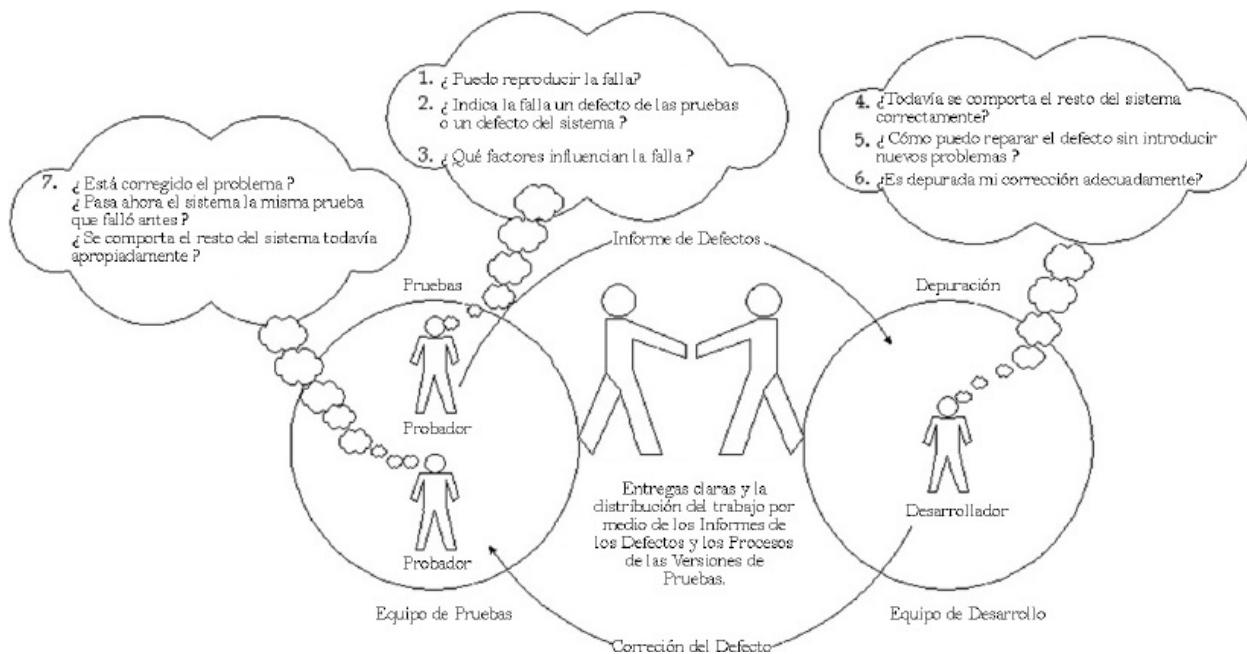


Figura 1.1: Encontrar-Depurar-Confirmar

En la figura 1.1 se puede observar una vista gráfica del proceso de pruebas que detecta una falla, los programadores depuran esa falla y los probadores realizan las pruebas de confirmación.

Específicamente, el probador lleva a cabo los pasos del 1 al 3 del proceso, mostrados en la parte superior del diagrama. A su vez, el probador comprueba para ver si el problema es intermitente o reproducible. De igual forma, el probador comprueba para distinguir entre una falla que surge de un defecto del sistema y una falla que surge debido a los casos de prueba, datos de prueba inadecuados y así sucesivamente. Así también, el probador comprueba para ver si diferentes valores de configuración o de datos podrían cambiar el comportamiento de la falla. Esta información ayuda al programador en la depuración, por lo tanto, el probador lo coloca en su informe de defectos.

Glosario del ISTQB

Caso de prueba: Un conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y posiciones de ejecución, desarrollados para un objetivo o condición de prueba particular, como para ejercitarse en un camino particular del programa o para verificar la conformidad con un requisito específico.

Luego el informe de defectos es entregado al equipo de desarrollo. Como se puede observar en la parte central de la figura 1.1, la entrega debería estar bien definida. De otra manera, hay un riesgo al que nosotros llamamos "pingpong". El informe de defectos podría ir y venir entre los dos grupos, siendo aclarado, antes que el programador pueda actuar en éste.

Una vez que el programador acepta el informe de defectos, asumiendo que el defecto debe ser corregido, él identifica la causa raíz. Él trata de repararlo sin producir más problemas. Luego él realiza la prueba unitaria de la corrección.

Esto nos conduce a otra entrega, a través del proceso de versiones de las pruebas. Sin embargo hablaremos acerca de esto más adelante al igual que de los informes de defectos, esto debe ser bien definido.

Glosario del ISTQB

Pruebas: El proceso que consiste en todas las actividades del ciclo de vida, tanto estáticas como dinámicas, interesadas en la planificación, preparación y evaluación de los productos de software y los productos del trabajo relacionados, para determinar que satisfagan los requisitos especificados, para demostrar que son aptos para el propósito y para detectar

defectos.

Finalmente el probador ejecuta las pruebas de confirmación y regresión contra el sistema con la corrección del defecto. Con optimismo, esto resulta en el cierre del informe. Sin embargo, puede ocurrir lo que llamamos “rizar el rizo”, en inglés “loop-the-loop”, cuando la corrección de un defecto no repara la falla completamente, resultando en una reapertura del informe.

Claramente ambos, el “ping-pong” y “rizar el rizo” son dañinos para la efectividad y eficiencia del equipo de pruebas, el equipo de desarrollo y todo el equipo del proyecto. Este ciclo encontrar-depurar-confirmar es un ciclo crítico, porque un proyecto podría recorrer decenas o incluso cientos de veces durante un proyecto. Tiene que ser realizado correctamente.

Más antes mencionamos un problema común para los grupos de pruebas, aquél de las expectativas desalineadas. Una de esas expectativas desalineadas ocurre cuando las personas piensan en las pruebas meramente como la ejecución de pruebas contra el sistema, a menudo justo en el final del proyecto.

Existen otras actividades importantes de las pruebas, así como:

- Planificar y controlar.
- Seleccionar las condiciones de las pruebas.
- Diseñar los casos de prueba.
- Comprobar los resultados de las pruebas.
- Evaluar los criterios de salida.
- Informar los resultados de las pruebas.
- Tareas del cierre o fin de las pruebas.

Estas actividades serán cubiertas más adelante en este libro.

1.2.1 Ejercicios

Ejercicio 1

Un programa acepta tres enteros que representan las longitudes de los lados de un triángulo⁷.

Éste da como resultado “escaleno” (lados no iguales), “isósceles” (dos lados iguales), o “equilátero” (tres lados iguales).

Escriba un conjunto de casos de prueba efectivos (que encuentren defectos comunes) y eficientes (tan pocas pruebas como sea posible). Usualmente un caso de prueba consiste en la acción del probador, el dato y el resultado esperado, pero aquí la acción es generalmente el “dato de entrada”.

Argumente.

Solución del Ejercicio 1

Este ejercicio es famoso, encontrado en el primer libro de pruebas de software de Glenford Myers *The Art of Software Testing*. Myers presenta este ejemplo aparentemente trivial y luego muestra muchas soluciones posibles para el mismo.

En la búsqueda de su propia solución, hágase algunas preguntas.

Primer, ¿Cubrió la entrada de datos válidos e inválidos? La mayoría de las personas que han sido probadores por un tiempo son buenas en probar con los inválidos y de hecho algunas veces enfatizan demasiado esas pruebas y no cubren importantes condiciones válidas.

Segundo, ¿Cuáles pruebas ejecutaría primero? A algunas personas les gusta empezar con las varias entradas inválidas, forzando muchas condiciones de error. Ese método está bien para una aplicación estable, pero para aplicaciones menos estables es usualmente mejor ejecutar primero algunos casos de prueba básicos válidos.

Analicemos nuestra solución. Para crearla, utilizamos una combinación del particionamiento de equivalencias y el análisis de valores límites.

Estamos interesados solamente en la funcionalidad de las pruebas y la interfaz del usuario. En los siguientes ejercicios, hablaremos acerca de otros riesgos para la calidad del sistema acerca de los cuales podríamos estar interesados.

Note que nuestra solución es sólo una de un gran conjunto de posibles respuestas correctas. Algunas reglas genéricas son encontradas en el libro de Myers para la determinación de un buen conjunto de pruebas. Myers proporciona también soluciones para este ejercicio. Desde que él escribió su libro en los días de los programas de mainframes de pantalla verde, él no tuvo que cubrir algunas de las pruebas de interfaz de usuario que hemos incluido.

<i>Partición de entradas erróneas</i>	
Un caso de prueba con a, b o c (un no numérico)	Un mensaje de error
Un caso de prueba con a, b o c (un número real)	Un mensaje de error
<i>Partición del Triángulo Escaleno</i>	
2, 3, 4	Escaleno
$\maxint-2, \maxint-1, \maxint$ (\maxint = Máximo entero permitido)	Escaleno
1, 2, 3	Un mensaje de error
5, 3, 2	Un mensaje de error
1, 2, 4	Un mensaje de error
-2, 3, 4	Un mensaje de error
0, 1, 2	Un mensaje de error
$\maxint-1, \maxint, \maxint+1$	Un mensaje de error
<i>Partición del Triángulo Isósceles</i>	
2, 2, 1	Isósceles
9, 5, 5	Isósceles
$\maxint, \maxint-1, \maxint$	Isósceles
1, 2, 1	Un mensaje de error
2, 5, 2	Un mensaje de error
3, 1, 1	Un mensaje de error
2, -2, 1	Un mensaje de error
2, 2, 0	Un mensaje de error
$\maxint, \maxint+1, \maxint$	Un mensaje de error
<i>Partición del Triángulo Equilátero</i>	
1, 1, 1	Equilátero
$\maxint, \maxint, \maxint$	Equilátero
$\maxint+1, \maxint+1, \maxint+1$	Un mensaje de error
1, 1, -1	Un mensaje de error

0, 0, 0	Un mensaje de error
<i>Pruebas de la Interfaz de Usuario</i>	
Desborde cada búfer de entrada para a, b y c	Un mensaje de error
Trate de adicionar un espacio delante de a, b y c	Un mensaje de error o ignora el espacio
Trate de adicionar un espacio después de a, b o c	Un mensaje de error o ignora el espacio
Deje a, b o c en blanco (uno a la vez)	Un mensaje de error

Tabla 1.1: Solución-Pruebas del Triángulo

Aquí tenemos alrededor de 40 pruebas. Sin embargo, pudimos reducir el número de pruebas a 21 si no tratamos con condiciones similares de error para las varias particiones. Hay dos suposiciones simples que hacen que esto funcione.

Primero, note que los tres enteros sólo pueden describir un triángulo cuando la suma de los dos lados es mayor que el tercer lado, para todas las combinaciones posibles. Probablemente podríamos terminar con seis pruebas en total para los posibles errores lógicos, tres para cada posible longitud de lado que podría ser mayor que la suma de los otros dos lados, tres por cada posible longitud de lado que podría ser igual a la suma de los otros dos lados.

Segundo, note que un programador inteligente escribiría una función que validaría que al programa le haya sido ingresado un entero y luego reutilizaría esa función para cada campo de entrada. Entonces, sólo tendríamos que probar un carácter, un número real, un cero, un entero negativo, un $\maxint+1$, un desborde de búfer, un espacio al comienzo, un espacio al final y un espacio en blanco, para nueve errores de entrada.

Entonces, ¿Cómo lo hizo? ¿Qué probamos nosotros que usted no probó? ¿Qué probó usted que nosotros no probamos?

Si usted no encontró algunos de estos, no se sienta mal. Al final de este libro, podrá identificar no sólo todas las pruebas de arriba, sino también pruebas en otras áreas de riesgo.

Sin embargo, aún cuando haya leído el libro completamente, podría ser necesario que tenga que consultar libros y notas para llegar a un conjunto completo de las pruebas. Tuvimos que volver a leer el desarrollo del ejercicio de Myers para finalizar nuestra solución.

Esto apunta a un tema importante de este libro. La realización del diseño e implementación de las pruebas sin material de referencia lleva a menudo a omisiones en las pruebas. La realización del diseño y la implementación de las pruebas con restricciones apretadas y estresantes respecto al tiempo que existen durante la ejecución de las pruebas, hacen que tales omisiones sean aún más probables, aún para probadores experimentados. Esto subraya la necesidad de un análisis, diseño y una implementación cuidadosa de las pruebas por adelantado, en vez de basarse solamente en las técnicas exploratorias o informales. Hay un lugar para las pruebas exploratorias en casi cada proyecto de pruebas, pero no como una técnica principal de pruebas.

1.3 Principios Generales de las Pruebas

Objetivos del Aprendizaje

LO-1.3.1 Explicar los siete principios de las pruebas. (K2)

Glosario del ISTQB

Pruebas exhaustivas: Un método de pruebas en el cual el juego de pruebas comprende de todas las combinaciones de los valores de entrada y las precondiciones.

Esta sección, Principios Generales de las Pruebas, explicará los siguientes principios de las pruebas:

- Las pruebas revelan la presencia de defectos.
- La imposibilidad de pruebas exhaustivas.
- Los beneficios de las pruebas tempranas.
- La aglomeración de defectos: el agrupamiento de defectos.
- La paradoja del pesticida.
- Las pruebas deberían adaptarse a necesidades específicas.
- La falacia de la ausencia de errores.



Figura 1.2: Las Pruebas Revelan la Presencia de Defectos: Una Metáfora

El primer principio, las pruebas revelan la presencia de defectos, es mejor explicado a través de una metáfora hortícola.

Imagine que tiene una huerta hermosa. Está particularmente orgulloso de sus tomates. Un día caminando a través de su huerta usted ve todas las ramas del tomate desprovistas de sus hojas, quedando sólo con sus tallos.

Siendo un jardinero experimentado, sabe lo que está pasando. "Oh no," piensa usted "¡tengo gusanos picudos!" A propósito, un gusano picudo, es el gusano desagradable que se muestra en la figura 1.2.

Muy bien, hasta esta parte usted sabe ciertamente que tiene insectos en su jardín. No necesariamente ha visto los gusanos picudos, sin embargo ha visto sus inconfundibles huellas.

Pero, considere esto: si no hubiera visto las huellas, ¿Podría estar seguro de que no tuviera insectos? Tal vez pudiera estar bastante seguro de que no tenía gusanos picudos, sin embargo ¿Qué hay de otros insectos?

Algunos insectos son fáciles de detectar otros no. Las pruebas son como caminar por su huerta. Esto puede revelar la presencia de insectos, pero no puede probar la ausencia de ellos.

Esto nos lleva al siguiente principio de las pruebas, el cual es que las pruebas exhaustivas son imposibles.

Desafortunadamente, mucha gente piensa que la misión del equipo de pruebas es, "Sólo asegurar que el software funcione antes de enviarlo". Este privilegio es completamente imposible por razones que son bastante fáciles de formular.

Por un lado si se considera el número de caminos diferentes de ejecución en cualquier parte de un software no trivial, ellos son casi infinitos. Recuerde, las aplicaciones modernas de software consisten en decenas o cientos de millones de sentencias. El flujo de control pasa a través de esas sentencias por cientos, miles o millones de decisiones. Cada una de esas decisiones puede muchas veces combinarse independientemente con todas las otras decisiones. Por ejemplo, un científico de computadoras de Sun Microsystems estimó que el número de posibles estados en un servidor Solaris era en el orden de magnitud más grande que el número de moléculas en el universo.

Por otro lado, no son sólo flujos de control. Los programas existen típicamente para manipular datos, muchas veces datos complejos. Entonces, tenemos también grandes y complejos flujos de datos en los programas. Algunos de estos flujos encaminan los datos dentro de otras áreas funcionales del sistema. Algunos de estos flujos encaminan los datos dentro de las bases de datos, de donde son entonces extraídos y utilizados más adelante.

Se pone peor. Porque el software no es construido de materiales estandarizados como un puente o avión, pequeños cambios en el sistema pueden causar regresiones significativas. Estas regresiones no sólo pueden ser lineales al tamaño de los cambios, sino incluso también pueden no estar obviamente relacionados al cambio.

Finalmente, los usuarios utilizan software de varias formas, algunas de las cuales son muy difíciles, sino imposibles de predecir. Los usuarios utilizan software en varias configuraciones de campo, algunas de las cuales ni siquiera existen en el momento de las pruebas.

Entonces, simplemente no hay forma de probar exhaustivamente, si por pruebas exhaustivas queremos decir, "Hasta que la probabilidad de descubrir un defecto sea igual a cero". Incluso con la definición de pruebas exhaustivas, que es bastante restrictiva en el glosario del ISTQB, dice que éstas prueban todas las combinaciones de las entradas y condiciones, lo cual todavía no es posible.

Eso dicho, estas expectativas de que uno puede probar exhaustivamente, o reducir la probabilidad de un defecto no descubierto a cero, son muy, muy comunes. De hecho, cuando hacemos evaluaciones para clientes, NO es raro encontrar esas expectativas equivocadas.

A su vez, estas expectativas equivocadas crean problemas para los profesionales y equipos de pruebas. Por un lado, crean demandas altas e inalcanzables en el grupo de pruebas. Por otro lado,

los grupos de pruebas operan típicamente como una organización de servicios, y las organizaciones de servicios son típicamente percibidas como incompetentes por sus clientes cuando las demandas—razonables o de otra forma—no son cumplidas.

Entonces, los probadores deben estar listos para comunicar cómo las pruebas pueden contribuir y calmar las expectativas equivocadas. Además, debemos utilizar palabras que los interesados del negocio del proyecto entenderán. Afortunadamente este libro le ayudará a aprender eso.

Otro principio de las pruebas es el beneficio del temprano aseguramiento de la calidad y las pruebas tempranas. En pocas palabras, este principio dice que el daño de los defectos aumenta entre más tiempo estén en el sistema.

Para ampliar esto, el costo de un defecto tiende a aumentar a medida que el proyecto continúa. El aumento es generalmente más multiplicativo que aditivo, es decir, tiende a incrementarse el doble o el triple o más de una etapa a la siguiente.

Además, dado que la mayoría de los costos asociados con los defectos de una pre-versión tienden a estar asociados con el esfuerzo necesario para eliminarlos, costos más altos significan cronogramas más largos. En otras palabras, el daño no es sólo financiero, sino también un daño al cronograma.

Finalmente, debido a que las actividades del aseguramiento de la calidad y de las pruebas tienden a actuar como un filtro, ignorando alguna proporción de defectos los cuales están presentes, mientras más defectos están presentes durante una actividad del aseguramiento de la calidad o de las pruebas, más defectos se escaparán de esa actividad.

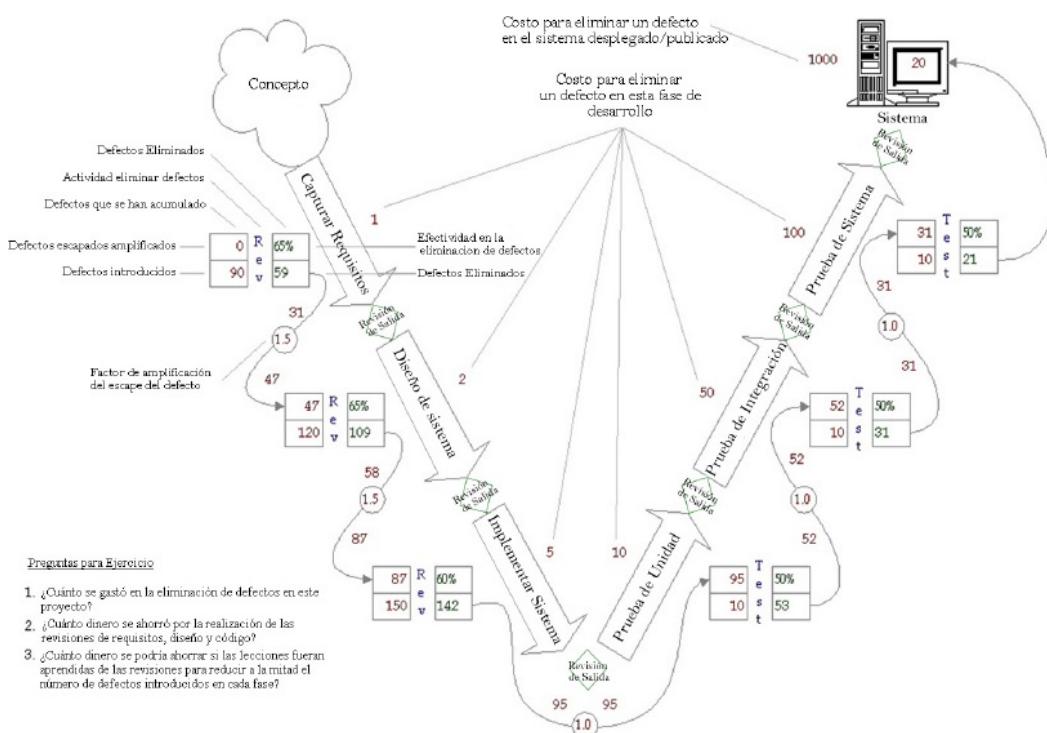


Figura 1.3: Beneficios del Temprano Aseguramiento de la Calidad y las Pruebas Tempranas

Esta figura tiene una gráfica muy rica en información que ilustrará el principio de los beneficios del temprano aseguramiento de la calidad y las pruebas tempranas. Recomendamos que estudie este gráfico por un momento antes de ver nuestra explicación acerca de éste. El gráfico es complejo, pero le llevará a una comprensión amplia y profunda de este principio.

En primer lugar, se ve que el ciclo de vida del desarrollo se presenta como una gran "V". En la parte superior izquierda, aparece el concepto del sistema. A través de la descomposición progresiva, perfeccionamos ese concepto en los requisitos, luego, en un diseño y finalmente en una implementación. Subiendo por el lado derecho de la V, realizamos tres niveles de pruebas, pruebas unitarias, pruebas de integración y prueba de sistema. En la parte superior derecha, entregamos el sistema a la producción.

En el interior de la "V", vemos los costos de la eliminación de un defecto en cada fase del ciclo de vida. Tenga en cuenta el aumento multiplicativo. Estos números son hipotéticos. Sin embargo, la magnitud del aumento de una fase a la siguiente es, en algún caso, conservadora.

En el exterior de la "V", vemos seis tablas. Cada tabla tiene dos columnas y dos filas. Las columnas de la izquierda se refieren a la acumulación e introducción de defectos. Las columnas de la derecha están relacionadas con la eliminación de defectos, tanto en porcentajes como en números absolutos.

En las cajas del lado izquierdo de la V, vemos "Rev." en el centro. Esto significa que las revisiones se utilizan para detectar y eliminar defectos. Concretamente las revisiones de requisitos, revisiones de diseño y revisiones de código.

En las cajas del lado derecho de la V, vemos "Prueba" en el centro. Esto significa que las pruebas se utilizan para detectar y eliminar los defectos. Concretamente pruebas unitarias, pruebas de integración y pruebas del sistema.

El número de defectos introducidos en cada fase es hipotético, pero proporcional a los promedios de la industria publicado por Capers Jones. El porcentaje (y por tanto el número) de defectos eliminados en cada fase también corresponden aproximadamente a los informes de Capers Jones.

También vemos que mientras los defectos se escapan de una fase a la siguiente, ellos son amplificados en la parte izquierda de la V. Esto significa que un defecto en los requisitos resulta más que un defecto en el diseño, y que un defecto en el diseño resulta más que un defecto en la implementación. Una vez que entramos en las fases de pruebas, no ocurren más amplificaciones de defectos.

Este diagrama así muestra el flujo de defectos a través del ciclo de vida. Los defectos son introducidos. Los defectos son detectados y eliminados. Los defectos se escapan a la siguiente fase.

Estudie este diagrama cuidadosamente. Se ilustra el costo de la calidad—es decir, el costo asociado con los defectos—para este proyecto hipotético. En el siguiente ejercicio que viene en breve, se le pedirá responder a tres preguntas:

- ¿Cuánto se gastó en la eliminación de defectos en este proyecto?
- ¿Cuánto dinero se ahorró por la realización de las revisiones de requisitos, diseño y código?
- ¿Cuánto dinero se podría ahorrar si las lecciones fueran aprendidas de las revisiones para reducir a la mitad el número de defectos introducidos en cada fase?

Piense un momento y vea si puede responder estas preguntas.

¿Listo? Bien, entonces abajo en la figura encontrará las respuestas a estas preguntas.

Fase	Pregunta 1 defectos			Pregunta 2 defectos			Pregunta 3 defectos		
	Presentes	Eliminados	Costo	Presentes	Eliminados	Costo	Presentes	Eliminados	Costo
Requisitos	90	59	59	90	0	0	45	29	29
Diseño	167	109	218	255	0	0	84	55	110
Implementación	237	142	710	533	0	0	119	71	355
Pruebas unitarias	105	53	530	543	272	2.720	53	27	270
Pruebas de integración	62	31	1.550	281	141	7.050	31	16	800
Prueba de sistema	41	21	2.100	150	75	7.500	20	10	1.000
Despliegue/entrega	20	20	20.000	75	75	75.000	10	10	10.000
Total			435 \$25.167			563 \$92.270			218 \$12.564

Figura 1.4: Solución del Ejercicio de la Figura 1.3

Como puede observar en esta figura presentamos la solución del ejercicio de la figura 1.3. Antes de explicarle las tres soluciones asumimos que el costo de la eliminación de un defecto en la fase de requisitos es de US\$ 1.

Solución de la primera pregunta: Se puede observar en la figura 1.3 que en la fase de requisitos se encontraron 90 defectos pero se corrigieron solamente 59 defectos (65% de efectividad), entonces el costo por ese esfuerzo es de $59 \times \$1 = \59 , luego en la fase de diseño se corrigieron 109 pero allí en esa fase el costo de la eliminación de defectos se duplica, por lo tanto el costo es $109 \times \$1 \times 2 = \218 , en la fase de la implementación el costo es de cinco veces más, $142 \times \$1 \times 5 = \710 , en la fase de pruebas unitarias el costo es de 10 veces más, $53 \times \$1 \times 10 = \530 , en la fase de pruebas de integración el costo es de 50 veces más $31 \times \$1 \times 50 = \1.550 , en la fase de pruebas de sistema el costo es de 100 veces más, $21 \times \$1 \times 100 = \2.100 y por último el costo en la fase producción/despliegue/entrega el costo es de 1.000 veces más, $20 \times \$1 \times 1.000 = \20.000 . Sumando los costos de cada fase nos da en total US\$ 25.167.

Solución de la segunda pregunta: Hemos visto que si realizamos las pruebas de revisión, requisitos y diseño entonces encontraríamos los defectos y los eliminaríamos como en la figura 1.3.

Para saber cuánto de dinero nos ahorraremos si no realizamos estas actividades entonces es necesario de suponer que al no realizar las revisiones ni en los requisitos ni en el diseño y ni en la implementación eso hace que dejemos todos los defectos acumulándose hasta después de la fase de implementación, primeramente dejamos que los 90 defectos en los requisitos se queden sin corregir y luego pasando a la fase de diseño se amplifican en número con el factor de 1.5 a $90 \times 1.5 = 135$ más los nuevos que aparecieron en la fase de diseño, es decir, $135 + 120 = 255$ defectos

acumulados sin corregir. Pasando a la fase de implementación los defectos acumulados también se amplifican con un factor de 1.5 a $255 \times 1.5 = 383$ defectos más los nuevos, 150, que aparecieron en esta fase, haciendo un total de $150 + 383 = 533$ defectos en total acumulados sin corregir hasta la fase de implementación y pasando a la fase de pruebas de unidad si lo seguimos dejando estos mismos defectos ya no se amplifican pero se acumulan junto con los nuevos 10 que aparecieran en esta misma fase haciendo un total de $10 + 533 = 543$ defectos acumulados sin corregir. Si ya comenzamos a corregir recién en esta fase que es la de pruebas unitarias, entonces nuestro costo sería un poco más costoso por que comenzamos tarde en el ciclo a encontrar y corregir todos estos defectos acumulados, supongamos que aquí corregimos sólo el 50% de los defectos que es usual según las estadísticas de Capers Jones, es decir corregimos 272 defectos solamente pero aquí el costo es $\times 10$ veces más por defecto a corregir, entonces el costo es de US\$ 2.272, luego pasando a las pruebas de integración no hay amplificación de defectos y adicionando los nuevos 10 que aparecieron entonces obtenemos $271 + 10 = 281$ defectos acumulados, de estos corregimos solamente un 50%, es decir 141 defectos con un costo de 50 veces más, $141 \times 50 = \text{US\$ } 7.050$, nos quedan 150 con los cuales pasamos a la fase pruebas de sistema, si corregimos allí con una efectividad del 50%, es decir corregimos sólo 75 defectos el costo se elevaría a 100 veces más por defecto, el costo sería de $75 \times \text{US\$ } 100 = \text{US\$ } 7.500$, nos quedan 75 defectos con los que pasaríamos a la fase de despliegue/producción/entrega, allí el costo por corrección de defecto es de $\times 1.000$ veces más, entonces el costo sería de $75 \times \text{US\$ } 1000 = \text{US\$ } 75.000$. Sumando los costos de corrección de defectos desde la fase de pruebas unitarias tendríamos un gasto total de US\$ 92.270.

Entonces para saber cuánto dinero ahorraremos al comenzar con las revisiones y corrección de defectos desde los requisitos, encontramos la diferencia entre US\$ 92.270 y US\$ 25.167, obtenemos US\$ 67.103, que es el valor neto que nos ahorraríamos al realizar las revisiones.

Solución de la tercera pregunta: Si aprendimos las lecciones de las revisiones anteriores, lo que nos haría reducir el número de defectos a la mitad en cada fase, eso significa que los defectos se reducen a la mitad al entrar en cada fase. Observemos la tabla de la pregunta 3 en la figura 1.4. Comenzamos con la mitad de los defectos es decir $90/2 = 45$ defectos, luego si corregimos esos defectos con una efectividad de aproximadamente un 65% entonces corregiríamos 29 defectos a US\$ 1 por defecto lo que hace un total de US\$ 29 de costos y nos quedarían $45 - 29 = 16$ defectos que luego esos se amplificarán al pasar a la siguiente fase con un factor de 1.5, $16 \times 1.5 = 24$ defectos, y adicionando la mitad de los defectos que aparecieron en esta fase esto sumaría $120/2 = 60$ defectos, $60 + 24$ defectos = 84 defectos, tenemos en total 84 defectos en esta segunda fase y si corregimos los mismos con una efectividad del 65% nuevamente corregiríamos solo 55 defectos a un costo del doble, es decir $55 \times \text{US\$ } 2 = \text{US\$ } 110$ costaría la eliminación de esos defectos. Y si seguimos de la misma manera hasta llegar a la fase de despliegue/producción/entrega obtendríamos un total gastado de US\$ 12.564 y se ahorraría un total de US\$ 25.167 - US\$ 12.564 = US\$ 12.603.

Otro principio general de las pruebas es el agrupamiento de defectos. Esto significa que los defectos no son distribuidos aleatoriamente y de manera uniforme en el sistema, sino que se agrupan.

Éste es un principio muy bien conocido que data de varias décadas. IBM, en estudios desde la década de los setenta, encontró que en su sistema operativo MVS, el 38% de los defectos ocurrían en el 4% de los módulos. Así mismo, para su base de datos IMS, el 57% de los defectos estaban en el 7% de los módulos.

Como una cuestión práctica, esto puede hacer que el mantenimiento de software sea una verdadera pesadilla. El experto en la industria del software Capers Jones informa que la presencia excesiva de módulos propensos a errores puede causar una reducción del 50% de la productividad en el mantenimiento del software.

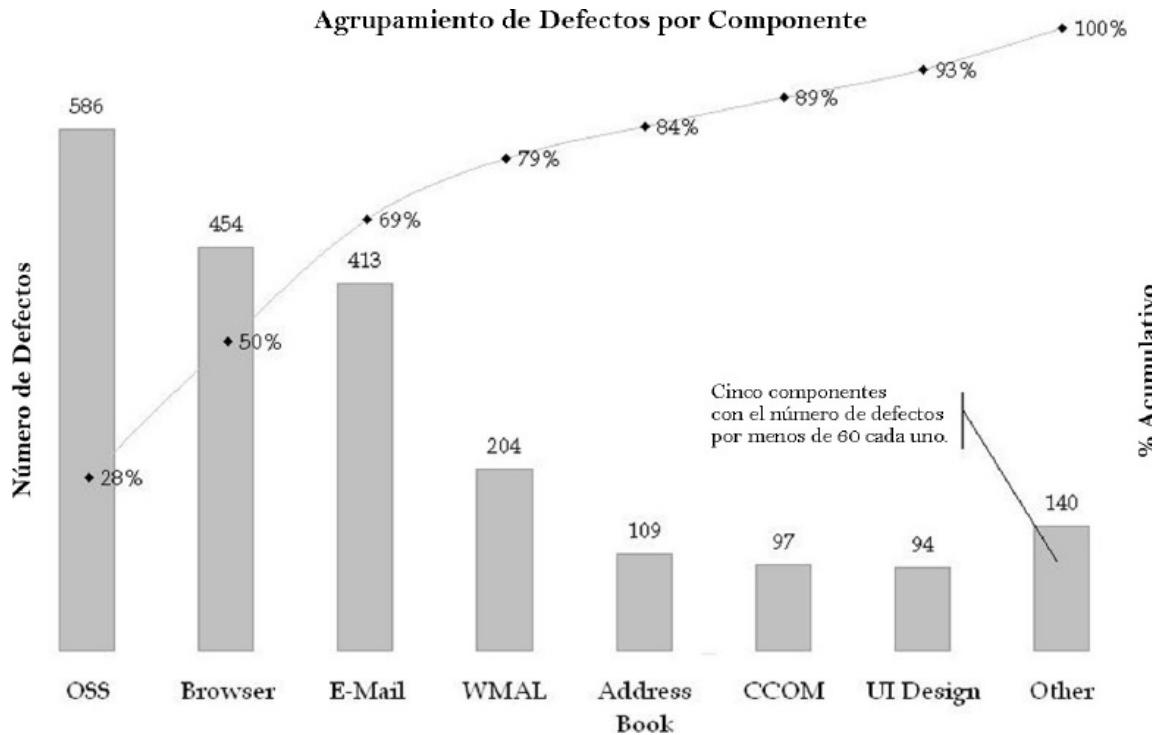


Figura 1.5: Agrupamiento de Defectos por Componente

En esta figura, usted puede observar un análisis que hicimos hace sólo unos pocos años atrás que muestra el mismo principio. El proyecto fue para un proyecto de un dispositivo de internet. Primero que nada, déjenos explicar este diagrama.

Este diagrama es llamado un diagrama de Pareto. En un diagrama de Pareto, usted crea un histograma que muestra el número de causas asociadas con un resultado. En este caso, la causa es el componente que contiene el defecto. El resultado es el número de defectos.

Un diagrama de Pareto ordena las causas desde las más frecuentes a la menos frecuente, con las causas más frecuentes al lado izquierdo. En la parte superior va un porcentaje acumulativo, el cual es el porcentaje de resultados asociados con la causa de abajo y a la izquierda.

En este diagrama, podemos ver que el 28% de los defectos estaban en el componente OSS. El 50% de los defectos estaban juntos en el Navegador y los componentes OSS; es decir, el otro 22% estaban en el navegador. Si considera los componentes OSS, el navegador y los de correo electrónico, ellos hacen más de los 2/3 de los defectos.

Estudie el diagrama cuidadosamente por un momento y podrá ver cuán desigual es la distribución de defectos.

Este análisis es realmente fácil de hacer, si puede extraer de su sistema de seguimiento de defectos los componentes implicados para cada defecto. Excel puede crear este diagrama para usted. Es casi seguro que verá el mismo tipo de distribución desigual. Esto ilustra, en un gráfico impactante, el principio del agrupamiento de los defectos. Inténtelo. Para nosotros es sorprendente cómo pocas organizaciones usan esta información para guiar sus pruebas, dado que son tan fáciles de obtener.

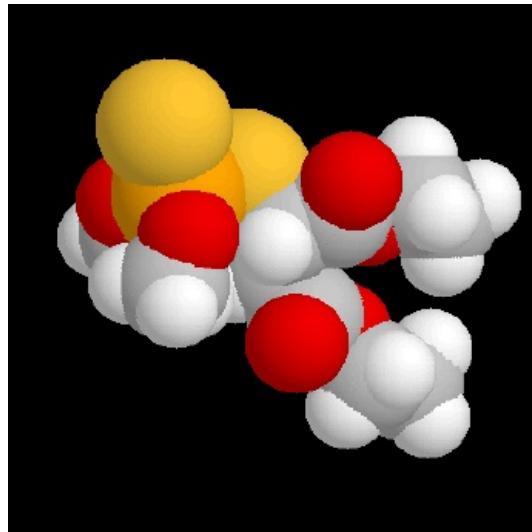


Figura 1.6: Paradoja del Pesticida

Otro principio de las pruebas es la llamada “paradoja del pesticida”. Para explicar esto, regresemos a nuestra metáfora de la huerta.

Recuerde que cuando nos fuimos, sus arbustos de tomate estaban sufriendo una infestación del gusano picudo. Entonces, supongamos que fumiga su huerta con pesticida, específicamente el material mostrado en esta figura, que es la estructura molecular del malatión⁸.

¡Qué bien! Los gusanos picudos mueren. Sin embargo, el malatión, si bien es un pesticida útil, no mata todos los insectos. Si continúa utilizando el malatión, los insectos que puede matar serán cada vez menos, mientras que los insectos que no puede matar no se verán afectados.

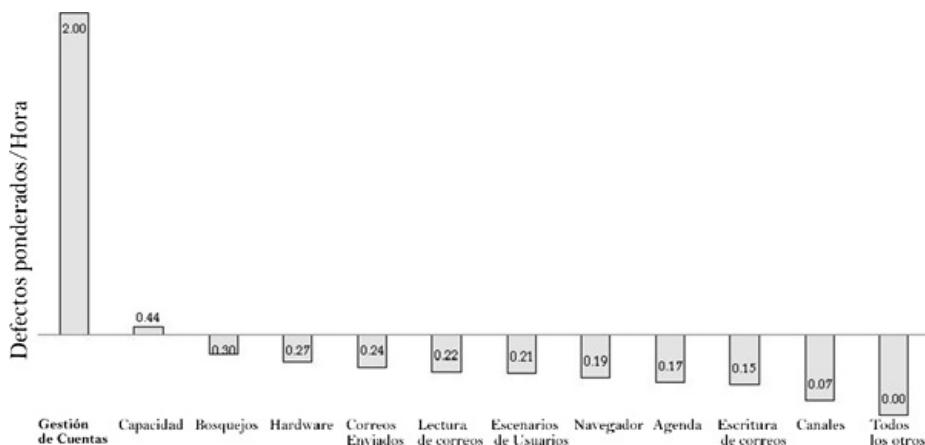
Así como los pesticidas se vuelven menos eficaces, también las pruebas. La ejecución de las mismas pruebas una y otra vez encontrará gradualmente menos defectos. La ejecución de un tipo de prueba no encontrará otros tipos de defectos. Por ejemplo, utilizando las pruebas funcionales no se encontrarán defectos de rendimiento.

Por lo tanto, tendrá que intentar con nuevas técnicas de pruebas—si el objetivo es encontrar defectos. Recuerde, las pruebas también tienen como objetivos la construcción de confianza, la reducción de riesgos y otros objetivos. Esos objetivos podrían ser bien atendidos por la ejecución de pruebas que no encuentran defectos.

Consideremos, por ejemplo, las pruebas de regresión. Las pruebas de regresión encuentran relativamente pocos defectos, pero sí construyen la confianza de que un determinado conjunto de cambios no ha dañado otras partes del sistema.

Examinemos cómo la paradoja del pesticida se muestra en un proyecto real, en la siguiente figura.

Estas dos imágenes comparan las eficiencias de los 11 mejores juegos de pruebas que encontraron defectos (de 27). La gráfica de arriba muestra la primera vez que el juego de pruebas ha sido ejecutado. La gráfica de abajo muestra los resultados para la quinta vez que el mismo juego de pruebas ha sido ejecutado. La eficiencia media del juego de pruebas en la primera pasada, 0,4, es utilizada como el punto de cruce de ejes en ambas gráficas. Con una excepción, los mismos juegos de pruebas están en los 11 mejores juegos de pruebas, los cuales muestran nuevamente la agrupación de los defectos. Sin embargo, todos los juegos de pruebas son menos efectivos después de cinco ejecuciones.



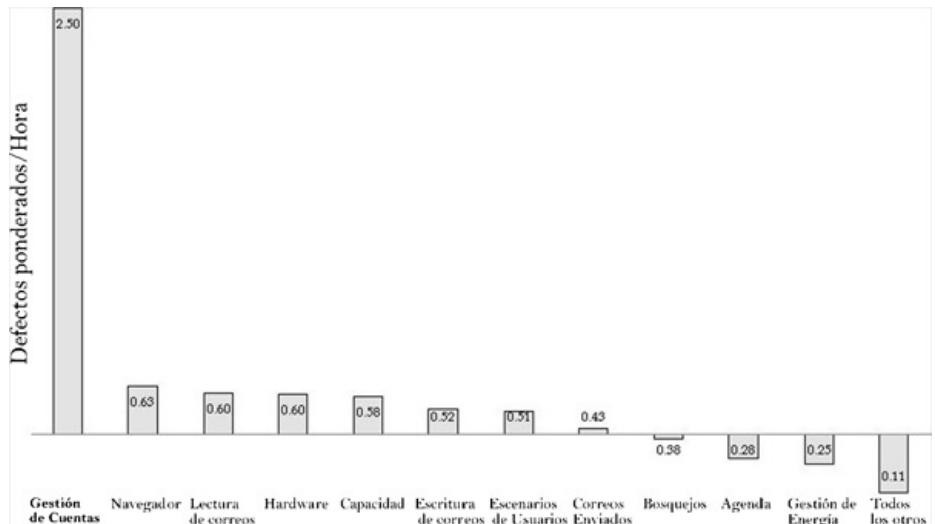


Figura 1.7: Histograma que Clasifica las Eficiencias de la Ejecución de Varios Juegos de Prueba

Bueno, esta figura muestra dos análisis idénticos. Cada análisis muestra un histograma que clasifica las eficiencias de la ejecución de varios juegos de prueba durante el proyecto de un dispositivo de Internet.

El resultado Defecto/Hora Ponderado para cada juego de prueba examina la tasa del hallazgo de los defectos, con los defectos ponderados por la severidad y prioridad. Observe que hemos normalizado por el tiempo de ejecución, el cual toma en cuenta el esfuerzo total distinto para los juegos de prueba.

En el proyecto del dispositivo de Internet, desarrollamos un conjunto de juegos de prueba, luego ejecutamos esos mismos juegos de prueba una y otra vez en un conjunto de ejecuciones de pruebas, corrigiendo poco a poco los defectos del sistema. Con el tiempo, los juegos de prueba encontraron cada vez menos defectos, como se evidencia en estos diagramas.

En cada análisis, se muestra una barra individual en el histograma para los 11 mejores juegos de prueba que encuentran defectos basados en esta métrica de eficiencia. Los otros 16 juegos menos eficientes son agrupados como "Todos los Demás" en la extrema derecha.

El gráfico superior muestra los puntajes de eficiencia para los juegos de prueba la primera vez que el conjunto de juegos de prueba ha sido ejecutado. El gráfico inferior muestra los resultados de la quinta vez que el mismo conjunto de juegos de prueba ha sido ejecutado.

Ahora, hemos calculado el promedio de la eficiencia del juego de prueba en la primera ejecución. Este promedio es de 0,4. Hemos utilizado este promedio como el punto de cruce de los ejes en ambos gráficos.

Estudie la gráfica cuidadosamente para asegurarse de que sabe cómo leerla.

Ahora, lo primero que observamos en este gráfico es el principio del agrupamiento de defectos que se muestra de nuevo. Puede ver, con una excepción, que los mismos juegos de prueba están en los 11 juegos de pruebas superiores. También se puede ver que la primera vez que ejecutamos las pruebas, ocho juegos de prueba están por encima del promedio de la eficiencia del hallazgo de los defectos, mientras que las otras 19 están por debajo del promedio.

La segunda cosa que notamos en este gráfico es la paradoja del pesticida. Después de la quinta ejecución, comparamos la eficiencia de cada juego de prueba contra el promedio de la eficiencia de los mismos durante la primera ejecución.

Aquellos que son más eficientes que el promedio de la eficiencia de la primera ejecución están por encima de la línea; aquellos menos eficientes están por debajo. Tenga en cuenta que sólo dos juegos de prueba son más eficientes en la quinta ejecución que el promedio de la primera ejecución. Además, si compara las eficiencias de cada juego de prueba para la primera ejecución contra la quinta ejecución, observará que todos los conjuntos de prueba son menos efectivos después de las cinco ejecuciones.

Otro principio de las pruebas es la falacia de la "ausencia de errores". Este principio establece que sólo encontrar y corregir muchos defectos, no garantiza al usuario, el cliente o el interesado del negocio la satisfacción del resultado.

Esto parece algo contra-intuitivo, pero es confirmado una y otra vez. Un montón de productos con pocos defectos han fracasado en el mercado al competir con otros productos con más defectos que sin embargo, tenían otros atributos que eran más importantes para los interesados del negocio.

Por ejemplo, consideremos la familia de sistemas operativos Unix, en comparación con el sistema operativo Windows. Usando las medidas más objetivas de la defectuosidad, Windows es

considerablemente mucho más defectuoso que cualquier variante de Unix. Sin embargo, Unix no es tan útil para los típicos usuarios no técnicos. Además, las variaciones innumerables que aparentemente existen, junto con el hecho de que algunas variantes han desaparecido, hace que los profesionales de TI sean reacios de utilizar Unix ampliamente.

Por lo tanto, aquí hay un punto clave: Los proyectos exitosos equilibran las fuerzas que compiten en cuanto a las características, el cronograma, el presupuesto y la calidad. Esto significa que debemos comprender que a veces, la versión temprana con defectos conocidos es mejor que la versión posterior con menos defectos.

El último principio de las pruebas es que las pruebas deben adaptarse al contexto, la situación en la cual son llevadas a cabo. Diferentes proyectos, organizaciones y productos tienen diferentes necesidades de las pruebas y tenemos que identificar y servir a esas necesidades para ser exitosos. Esto es sentido común, pero, como dijo Mark Twain, el sentido común es desafortunadamente no muy común.

Algunas personas llevan este principio a los extremos, y piensan que cada proyecto es completamente único. Se obsesionan con lo que es específico, lo que los ciega a la posibilidad de aplicar las buenas ideas de otros proyectos, organizaciones y productos. No se equivoque. Existen las mejores prácticas de pruebas. Hablaremos de ellas en este libro. Sin embargo, cada una de estas mejores prácticas es una práctica general que debe ser adaptada a su proyecto. En algunos casos, por razones específicas, una buena práctica no aplicará a un proyecto particular. Nosotros le ayudaremos a reconocer esas situaciones en este libro.

Esta necesidad de aplicar con inteligencia y adaptar las mejores prácticas de pruebas es crítica. El fracaso de adaptar el equipo de pruebas y sus métodos a las necesidades específicas que están a la mano es una razón común para la disolución de los equipos de pruebas. Una vez más, las pruebas son una organización de servicios, por lo general, lo que significa que los servicios proporcionados deben ser los servicios necesitados.

1.3.1 Ejercicios

Ejercicio 1

Haga memoria acerca de un proyecto reciente.

Haga una lista de los principios de las pruebas que recuerda que hayan sido vistos.

Haga una lista de los principios de las pruebas que recuerda que no hayan sido vistos.

Argumente.

Solución del Ejercicio 1

La mayoría de la gente se da cuenta de que las pruebas dependen del contexto y tienen que adaptarse a las realidades. De hecho, si existe alguna, la gente tiende a llevar este principio a los extremos y utilizar la frase "somos diferentes" como un llamado para los atajos y violaciones de las mejores prácticas. Tiene que adaptar su método de pruebas a cada proyecto, pero recuerde que muchas reglas generales se aplican.

Un tema común es el exceso de confianza en las pruebas de la etapa final como una panacea de la calidad en los proyectos en los que hemos trabajado, con los equipos que hemos realizado consultoría y la gente que hemos entrenado. Esto viola tres principios de las pruebas: las pruebas demuestran la presencia de defectos (pero no demuestran su ausencia); las pruebas tempranas ayudan a prevenir defectos y ahorrar dinero; y la falacia de la ausencia de errores (es decir, el intento de probar la calidad del sistema al final).

Algunos de los otros principios de las pruebas pueden haber sido operables pero no vistos. Por ejemplo, el agrupamiento de defectos ocurre ya sea si sabe acerca de ello o no. Puede encontrar los agrupamientos de defectos utilizando un sistema de seguimiento de defectos y un análisis de causa raíz para reunir datos acerca de los defectos.

1.4 Proceso de Pruebas Básico

Objetivos del Aprendizaje

LO-1.4.1 Recordar las cinco actividades de pruebas básicas y las tareas respectivas desde la planificación hasta el cierre de pruebas. (K1)

Esta sección, Proceso de Pruebas Básico, cubrirá los siguientes conceptos clave:

- Actividades de planificación y control en el proceso de pruebas.
- Actividades de análisis y diseño de pruebas en el proceso de pruebas.
- Actividades de implementación y ejecución de pruebas en el proceso de pruebas.
- La evaluación de los criterios de salida de las pruebas y la creación de informes de los resultados de las pruebas en el proceso de pruebas.

- Actividades de cierre de pruebas en el proceso de pruebas.

Glosario del ISTQB

Pruebas de confirmación: Véase *repetición de pruebas*.

Repetición de pruebas: Pruebas que ejecutan los casos de prueba que fallaron la última vez que se ejecutaron, con el fin de verificar el éxito de las acciones correctivas.

Criterios de salida: El conjunto de condiciones genéricas y específicas, acordadas con los interesados del negocio, para permitir que un proceso sea finalizado oficialmente. El propósito de los criterios de salida es prevenir que una tarea sea considerada por finalizada cuando todavía hay partes pendientes de la tarea que no han sido finalizadas. Los criterios de salida se utilizan para planificar e informar contra la cesación de las pruebas.

Incidencia: Cualquier evento que ocurre y necesita investigación.

Pruebas de regresión: Pruebas de un programa previamente probado a raíz de una modificación para garantizar qué defectos no han sido introducidos o descubiertos en áreas no modificadas del software, como resultado de los cambios realizados. Se realizan cuando el software o su entorno son modificados.

Base de prueba: Todos los documentos de los cuales los requisitos de un componente o sistema pueden ser deducidos. La documentación acerca de la cual los casos de prueba se basan. Si un documento puede ser enmendado, sólo por medio de un procedimiento de enmienda formal, entonces la base de pruebas se denomina base de pruebas congelada.

Condición de prueba: Un ítem o evento de un componente o sistema que pudiera ser verificado por uno o más casos de prueba, p.ej., una función, transacción, característica, atributo de calidad o un elemento estructural.

Cobertura de pruebas: Véase *cobertura*.

Cobertura: El grado, expresado como porcentaje, hasta el cuál un ítem de cobertura especificado ha sido ejercido por un juego de pruebas. Tenga en cuenta que este término no ha sido referenciado específicamente en esta sección, pero se lo incluye aquí, porque es un sinónimo de cobertura de pruebas.

Datos de prueba: Datos que existen (por ejemplo, en una base de datos) antes que una prueba sea ejecutada, y que afectan al componente o sistema sometido a pruebas o son afectados por estos.

Ejecución de pruebas: El proceso de ejecución de una prueba en el componente o sistema sometido a pruebas, produciendo resultado(s) real(es).

Registro de pruebas: Un registro cronológico de detalles relevantes acerca de la ejecución de las pruebas.

Plan de pruebas: Un documento que describe el alcance, el método, los recursos y el cronograma de las actividades de pruebas previstas. Identifica entre otros ítems de pruebas, las características que tienen que ser probadas, las tareas de pruebas, quien hará cada tarea, el grado de independencia del probador, el entorno de pruebas, las técnicas de diseño de pruebas y los criterios de entrada y salida que serán utilizados, y las razones para su elección, y algunos riesgos que requieren planes de contingencia. Es un registro del proceso de planificación de pruebas.

Procedimiento de prueba: Véase *especificación del procedimiento de prueba*.

Especificación del procedimiento de prueba: Un documento que especifica una secuencia de acciones para la ejecución de una prueba. También conocido como guión de prueba o guión de prueba manual. Tenga en cuenta que este término no ha sido referenciado específicamente en esta sección, pero se lo incluye aquí, porque es un sinónimo de procedimiento de prueba.

Política de pruebas: Un documento de alto nivel que describe los principios, métodos y objetivos principales de la organización con respecto a las pruebas.

Estrategia de pruebas: Una descripción de alto nivel de los niveles de pruebas que deben ser realizados y las pruebas en esos niveles para una organización o programa (uno o más proyectos).

Juego de prueba: Un conjunto de varios casos de prueba para un componente o sistema sometido a pruebas, donde la poscondición de una prueba es utilizada a menudo como una precondition para la siguiente.

Informe del resumen de pruebas: Un documento que resume las actividades de las pruebas y los resultados. También contiene una evaluación de los ítems de las pruebas correspondientes contra los criterios de salida.

Testware: Artefactos producidos durante el proceso de pruebas necesarios para planificar, diseñar y ejecutar las pruebas, así como la documentación, los guiones, las entradas, los resultados esperados, los procedimientos de instalación y ajuste, los archivos, las bases de datos, el entorno y cualquier software adicional o utilitarios utilizados en las pruebas.

Criterios de entrada: El conjunto de condiciones genéricas y específicas para permitir que un proceso continúe con una tarea definida, p.ej. la fase de pruebas. El propósito de los criterios de entrada es evitar que una tarea comience, lo cual implicaría más esfuerzo (malgastado) en comparación con el esfuerzo necesario para eliminar los criterios de entrada fallados.

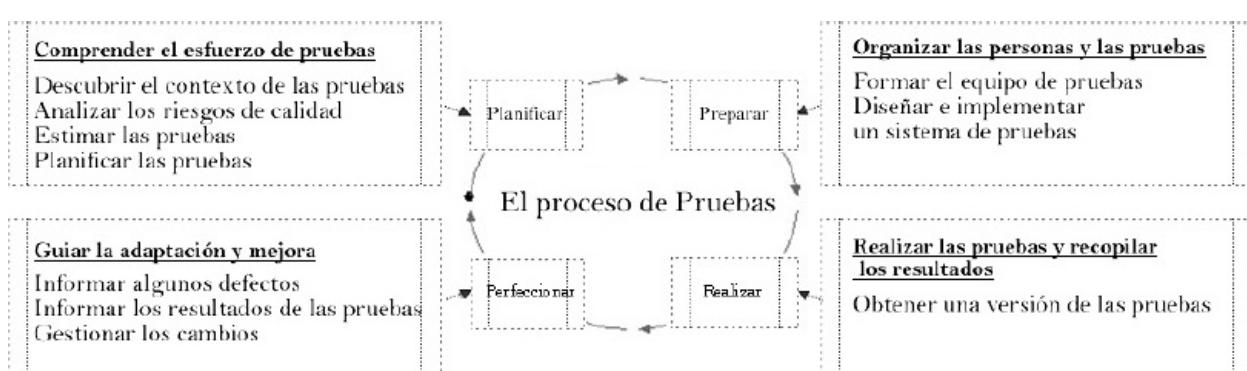


Figura 1.8: Los Procesos Críticos de las Pruebas

Hay una gran variedad de frameworks⁹ de procesos de pruebas de donde elegir. Se incluyen entre ellos el framework de los Procesos de Pruebas Críticos de RBCS, junto con las Pruebas de Software y el Proceso de Evaluación de SQE, T-MAP de Sogeti, el Mejoramiento de Procesos de Pruebas de Polteq y el Modelo de Madurez de Pruebas de Burnstein.

Los Procesos de Pruebas Críticos, las Pruebas de Software y el Proceso de Evaluación y T-MAP son no prescriptivos, porque proporcionan para el uso del valor del negocio la selección de qué proceso debe ser implementado u omitido, y cuales procesos tienen que ser mejorados y en qué orden. El Mejoramiento de Procesos de Pruebas y el Modelo de Madurez de Pruebas son prescriptivos, porque proyectan puntajes de madurez acerca de los frameworks y requieren que las mejoras sean implementadas en un orden particular a pesar de esos niveles de madurez.

El mundo real expone aun más variación, mientras las personas se enfocan en las pruebas con mayor o menor grado de formalidad y rigor.

El programa de estudios del ISTQB nivel básico propone un framework genérico del proceso de pruebas. Éste es no prescriptivo, porque no impone un esquema de madurez y reconoce el potencial de omisión de algunos elementos del framework en algunas circunstancias.

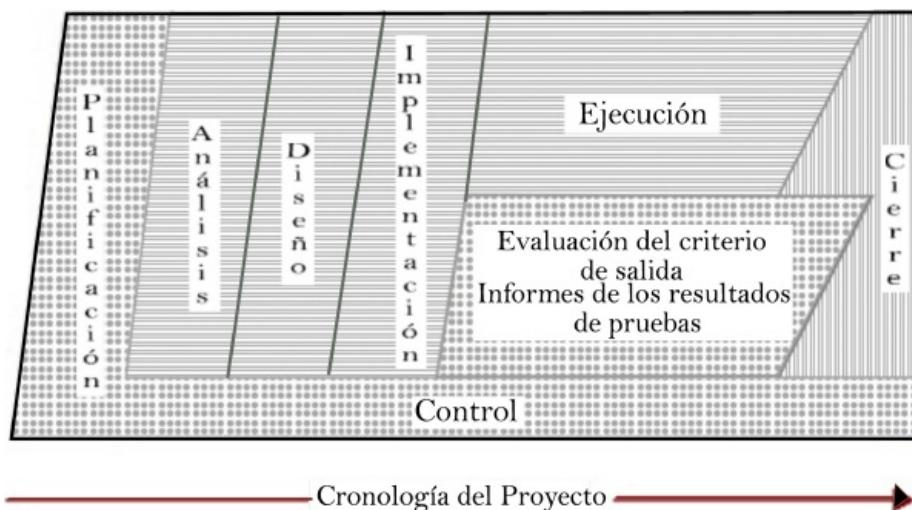


Figura 1.9: El Proceso de Pruebas Básico del ISTQB

El proceso de pruebas básico del ISTQB incluye los siguientes pasos:

- Planificación y control.
- Análisis y diseño.
- Implementación y ejecución.
- Evaluación de los criterios de salida de las pruebas y la creación de informes.
- Cierre de pruebas.

Dado que el proceso de pruebas tiene que ser adaptado al ciclo de vida del software, el proceso permite la superposición, la concurrencia e incluso la iteración en estos pasos.

La figura 1.9 muestra una posible instancia del proceso de pruebas básico del ISTQB. Las actividades que son principalmente de gestión están representadas por el área de puntos, mientras éas que son principalmente actividades de contribuyente individual están representadas por el área de líneas horizontales. El cierre, que es tanto de gestión como individual, está representado por el área de líneas verticales.

La distribución del tiempo es consistente con un proyecto de ciclo de vida secuencial, el cual se abordará más adelante.

En esta sección, revisaremos más de cerca a cada uno de estos pasos, incluyendo la descripción de las principales actividades que componen cada paso. Estas actividades y las tareas que las componen, serán descritas, con frecuencia en detalle, en los capítulos del 2 al 6 de este libro.

Empecemos con la planificación y el control, los cuales, por supuesto consisten principalmente en actividades de gestión.

La planificación incluye las siguientes actividades:

- Determinar el alcance de las pruebas, los riesgos, los objetivos y las estrategias.
- Determinar los recursos de las pruebas necesarios.
- Implementar las estrategias de las pruebas.
- Crear un cronograma del análisis y el diseño de las pruebas.
- Crear un cronograma de la implementación, la ejecución y la evaluación de las pruebas.
- Determinar los criterios de salida de las pruebas. El control incluye las siguientes actividades:
 - Medir y analizar los resultados.
 - Monitorear y documentar el progreso, la cobertura y los criterios de salida de las pruebas.
 - Iniciar acciones correctivas.
 - Tomar decisiones.

Tenga en cuenta que muchas actividades de planificación y control implican la obtención del acuerdo, el soporte y el consenso del equipo del proyecto y la gerencia del proyecto.

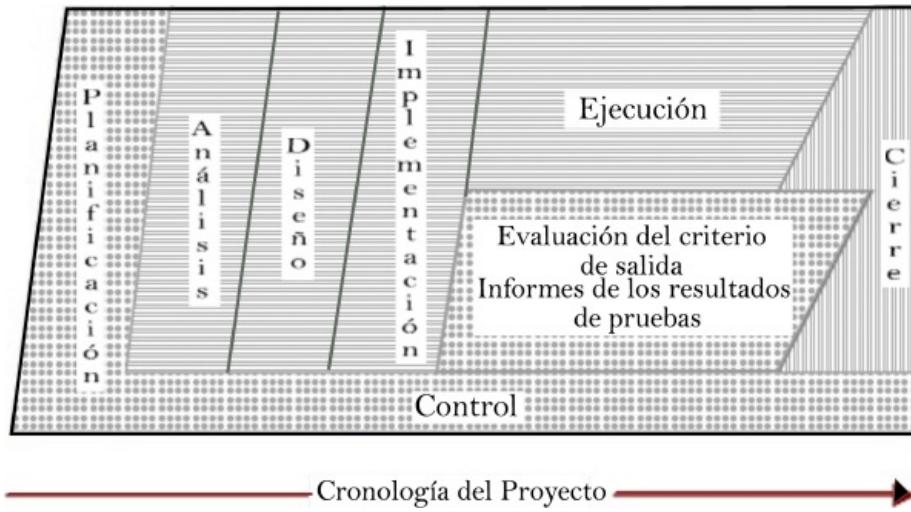


Figura 1.10: Análisis y Diseño

Pasemos ahora al análisis y el diseño de pruebas, que consisten principalmente en actividades de contribuyente individual.

El análisis incluye las siguientes actividades:

Revisar la base de pruebas. La base de pruebas es aquella en la cual las pruebas se basan, con frecuencia, incluyendo los requisitos o las especificaciones de diseño, las arquitecturas de red o sistema o los riesgos de calidad.

Identificar y priorizar las condiciones de pruebas, los requisitos de pruebas o los objetivos de pruebas y los datos de prueba necesarios. Esto requerirá a menudo un análisis de los ítems de pruebas, así como su comportamiento, especificación y estructura. Por supuesto, estos ítems podrían no existir todavía, entonces usted podría estar dependiendo de sus descripciones, así como los casos de uso.

Evaluar la comprobabilidad¹⁰ (“testability”) de los requisitos y el sistema. Esto podría resultar en la creación de informes acerca de algunos asuntos acerca de estas áreas para la gerencia.

El diseño incluye las siguientes actividades:

- Diseñar y priorizar combinaciones específicas de datos de prueba, acciones y resultados esperados para cubrir la base de pruebas, los riesgos importantes de calidad, y cualquier otra cosa que necesite cobertura.
- Identificar los datos de prueba necesarios para las condiciones y los casos de prueba.
- Diseñar del entorno de pruebas.
- Identificar alguna infraestructura y algunas herramientas necesarias.

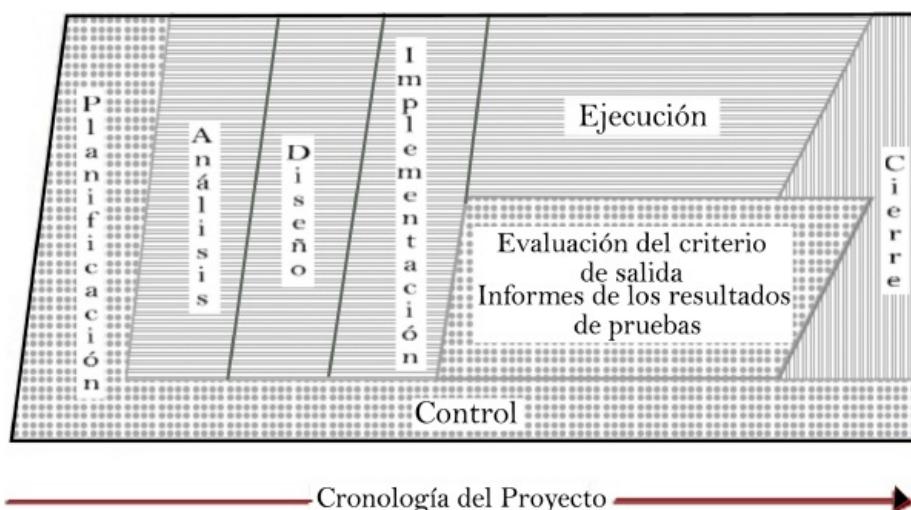


Figura 1.11: Implementación y Ejecución

Continuemos con la implementación y ejecución de las pruebas, las cuales, además consisten principalmente de las actividades de un contribuyente individual.

La implementación incluye las siguientes actividades:

- Desarrollar, implementar y priorizar casos de prueba, crear datos de prueba y escribir procedimientos de prueba.

- Preparar arneses de prueba y escribir scripts de pruebas automatizadas.
- Organizar juegos de prueba y secuencias de procedimientos de prueba para la ejecución eficiente de las pruebas, teniendo en cuenta las diversas restricciones que podrían determinar el orden en el cual las pruebas deben ser ejecutadas.
- Verificar que el entorno de pruebas ha sido instalado correctamente.

La ejecución incluye las siguientes actividades:

- Ejecutar casos de prueba tanto manuales como automatizados.
- Registrar los resultados de las pruebas, incluyendo las versiones del software sometido a pruebas, las herramientas de pruebas y el testware.
- Comparar los resultados reales y esperados, lo cual podría requerir la identificación de anomalías donde los resultados reales y esperados no coinciden.
- La investigación de anomalías puede resultar en la creación de informes y el análisis de las incidencias.
- Repetir las pruebas corregidas y/o actualizadas donde sea necesario.
- Ejecutar las pruebas de confirmación y/o regresión, cuando las nuevas versiones de pruebas lleguen.

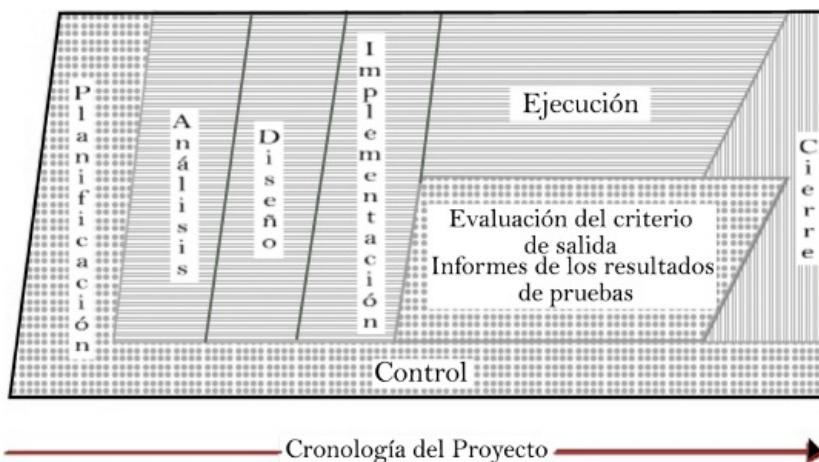


Figura 1.12: Criterios de Salida, Generación de Informes y Cierre

La evaluación de los criterios de salida y la creación de informes de los resultados de las pruebas están fuertemente superpuestas con la ejecución de las pruebas, las cuales consisten principalmente en actividades de gestión de actividades de gestión, incluyendo las siguientes:

- Comprobar los registros de las pruebas contra los criterios de salida de las pruebas especificados durante la planificación de pruebas.
- Evaluar si son necesarias más pruebas o si los criterios de salida especificados deben ser modificados.
- Escribir un informe del resumen de las pruebas para los interesados del negocio.
- A medida que la ejecución de las pruebas llega a un cierre, los criterios de salida han sido cumplidos y los informes finales de los resultados de las pruebas son recopilados, las actividades del cierre comienzan a ocurrir, lo cual consiste en actividades de gestión y actividades de contribuyente individual, incluyendo:
 - Confirmar los entregables de pruebas, la resolución final o postergación de los informes de defectos y la aceptación del sistema por las partes que lo reciben.
 - Finalizar y archivar el testware, el entorno de pruebas y la infraestructura de pruebas para su uso posterior durante el mantenimiento.
 - Entregar el testware y la posibilidad de ítems adicionales a la organización de mantenimiento
 - Realizar un estudio retrospectivo para tomar en cuenta las mejoras para las futuras versiones, los proyectos y los procesos de pruebas.

1.4.1 Ejercicios

Ejercicio 1

Haga memoria en un proyecto reciente.

Note cuáles de los pasos y tareas del proceso de pruebas del ISTQB fueron llevados a cabo.

Note cuáles de los pasos y tareas del proceso de pruebas del ISTQB no fueron llevados a cabo.

Note si algunos pasos se superpusieron o fueron completamente paralelos.

Argumente.

Solución del Ejercicio 1

No podemos hablar acerca de su último proyecto, pero hemos visto muchos patrones en el uso correcto e incorrecto de estos pasos del proceso de pruebas en los proyectos en los que hemos trabajado y realizado consultoría, y nos hemos basado en las centenas de argumentaciones en clase acerca del proceso de pruebas.

Es común que la planificación y el control, junto con el análisis y el diseño, sean realizados incorrectamente, en el último minuto o nada en absoluto. Por supuesto, mientras la planificación ocurre idealmente al principio de un proyecto, el control superpone todas las etapas del esfuerzo de las pruebas.

En algunos casos, el análisis es olvidado y la gente salta directo en el diseño de pruebas, lo cual resulta usualmente en omisiones significativas de cobertura. En otros casos, el análisis y el diseño son realizados en paquetes pequeños superponiéndose en el proyecto. Éste es un método que funciona muchas veces. Si planifica un método más secuencial para el análisis y diseño, recuerde que el diseño de pruebas conducirá usualmente a ajustes para el análisis.

En el caso de la implementación y ejecución, estos son usualmente realizados—por supuesto, sin ellos, no hay pruebas—pero no es inusual para ambos que sean reducidos prematuramente. Hemos tenido que posponer los esfuerzos de automatización importantes—in los cuales habíamos invertido literalmente meses-hombre de trabajo y que eso estaba a un par de semanas de ser completado— indefinidamente porque la dirección del proyecto sintió que las pruebas tenían que empezar inmediatamente. Algunas veces esas decisiones tienen sentido a nivel del proyecto, pero a menudo son el resultado de la prisa.

En cuanto a la evaluación del criterio de salida y creación de los informes de pruebas, los mejores proyectos en los que hemos trabajado han tomado seriamente los criterios de salida y han escuchado cuidadosamente a los informes del estado de las pruebas. Eso no quiere decir que el equipo del proyecto siguió siempre nuestras recomendaciones; tendemos a ser un jefe conservativo, en contra de los riesgos, mientras que muchos jefes de proyectos se sienten cómodos aceptando niveles de riesgos más altos. Sin embargo me desconcierta cuando los jefes de proyectos ignoran los hallazgos de las pruebas y no tienen criterios objetivos para la versión. ¿Qué sentido tiene realizar las pruebas en absoluto si no utiliza la información valiosa y ganada para guiar el proyecto al éxito?

De todas las partes del proceso del ISQTB, las actividades del cierre de pruebas, a menudo son las más saltadas o truncadas. En algunos casos, esto tiene sentido. Sin embargo, cuando un proyecto siguiente—así como un proyecto de mantenimiento—tiene que tratar de reutilizar los casos de prueba y otros entregables de pruebas, algún esfuerzo es esencial para conducir las pruebas a una conclusión ordenada.

1.5 La Psicología de Pruebas

Objetivos del Aprendizaje

LO-1.5.1 Recordar los factores psicológicos que influencian el éxito de las pruebas. (K1)

LO-1.5.2 Comparar la mentalidad de un probador y un desarrollador. (K2)

Glosario del ISTQB

Predicción de error: Una técnica de diseño de pruebas, donde la experiencia del probador se utiliza para anticipar qué defectos podrían estar presentes en el componente o sistema sometido a pruebas como consecuencia de los errores cometidos y para diseñar las pruebas específicamente para exponerlas.

Independencia: Este término no está definido en el glosario, pero es un término relacionado con la independencia de pruebas.

Independencia de pruebas: Consiste en la separación de las responsabilidades, lo que fomenta el logro de pruebas objetivas.

Esta sección, La Psicología de Pruebas, cubrirá los siguientes conceptos clave:

- Los factores psicológicos clave para el éxito de las pruebas.
- La importancia de objetivos claros.
- Los aspectos psicológicos de las autopruuebas¹¹ versus las pruebas independientes.
- La importancia de la comunicación respetuosa al criticar el trabajo de otros.
- Las perspectivas del programador y probador para el éxito.

Un buen probador tiene el conjunto correcto de los rasgos de personalidad y las inclinaciones mentales. Los atributos de un buen probador incluyen:

- Curiosidad.
- Pesimismo profesional.
- Un ojo crítico.
- Atención al detalle.
- Buenas habilidades de comunicación.

Por supuesto, la combinación apropiada de las habilidades es necesaria también.

¿Entonces, cuáles son estas habilidades más difíciles que un probador debe tener?

Bueno, por supuesto, un probador debe tener las habilidades profesionales básicas como la lectura y la escritura. La lectura es importante porque los probadores pasan mucho tiempo analizando las especificaciones, leyendo y respondiendo los mensajes de correo electrónico, revisando los casos de prueba de los demás y así sucesivamente. La escritura es importante porque los probadores pasan tiempo escribiendo los casos de prueba, los informes de los defectos, los informes del resumen de las pruebas y varios otros documentos relacionados con las pruebas.

Más allá de estas habilidades profesionales básicas, buscamos un probador que tenga habilidades en tres áreas principales: la tecnología, el dominio del negocio de la aplicación y las pruebas:

Las habilidades tecnológicas incluyen una comprensión básica de los lenguajes de programación, los sistemas operativos, las redes, HTML y las tecnologías Web o cualquiera de las otras tecnologías que son utilizadas para realizar el sistema sometido a pruebas. Algunas personas dicen que los probadores no deberían saber estas cosas, porque les haría aceptar muchos defectos como irrelevantes. La adaptación a comportamientos defectuosos es un verdadero problema para los probadores, pero no tiene nada que ver con la comprensión de la tecnología, sino más bien con la exposición repetida a los defectos y la orientación incorrecta de gestión.

Las habilidades del dominio de la aplicación están relacionadas con la banca, los factores humanos, las aplicaciones de escritorio o cualquier área de negocios o problemas que la aplicación está tratando de resolver. Básicamente, éstas son habilidades de los usuarios y los analistas de negocios, y los usuarios antiguos y los analistas de negocios pueden llegar a ser buenos probadores.

Por último, en el área de las pruebas, es importante que la gente reconozca a las pruebas como un área profesional especial con su conjunto único de habilidades. Éstas incluyen la escritura de guiones, el diseño de pruebas, la habilidad de explorar y atacar un sistema, la creación y la utilización de herramientas de pruebas, la construcción de modelos de rendimiento, la redacción de buenos informes de defectos y más.

Como una regla general, las organizaciones involucradas en la creación de software para el uso interno tienden a poner demasiado énfasis en el conocimiento del dominio. Las organizaciones que venden software tienden a poner demasiado énfasis en los conocimientos técnicos. La mayoría de las organizaciones tienden a subestimar el conocimiento de las pruebas.

Un buen jefe de pruebas gestionará y optimizará activamente las habilidades de su equipo. Un probador inteligente que prevé una carrera en este campo gestionará y optimizará activamente sus propias habilidades.

La mentalidad del probador es sutilmente diferente a la de otros en un equipo de proyecto.

Probar aplicaciones y revisar especificaciones son actividades diferentes que desarrollar código o elaborar requisitos de negocio. Las mentalidades requeridas para el éxito son también diferentes.

Esto no quiere decir que los programadores no puedan probar su propio código. Ellos pueden y deberían realizar las pruebas unitarias, y típicamente deberían participar en las pruebas de integración. Sin embargo, la mayoría de los programadores tienen la mentalidad incorrecta para realizarlo efectivamente. Ellos buscan la confirmación de que lo que han construido funciona, lo cual tiende a limitar psicológicamente su efectividad en el hallazgo y la eliminación de los defectos.

En algunos campamentos, concretamente algunos profesionales de las diversas metodologías ágiles, afirman: "los programadores que practican las pruebas unitarias automatizadas por medio de las Pruebas Dirigidas por Diseño, o algo así, no necesitan a los probadores cerca. Veamos algunos datos.

Capers Jones en sus estudios de miles de proyectos a través de cientos de organizaciones, informa que las pruebas unitarias, en promedio, tienen una efectividad de eliminación de los defectos de alrededor del 25%. Las mejores prácticas en las pruebas unitarias resultan en una efectividad de la eliminación de los defectos de alrededor del 50%. Cuando nuestras compañías, Business Innovations/RBCS, realizan evaluaciones de los equipos de pruebas de los clientes, encontramos **rutinariamente** una efectividad de la detección de los defectos del 80% al 85%, con un 90% hasta el 95%, que no es en absoluto inusual.

Ahora, estamos seguros de que hay múltiples razones de por qué existe esta gran diferencia en la eficacia de la detección de los defectos. Una de esas razones es que la separación de las funciones (a través de un equipo de pruebas independiente) ayuda a centrar las pruebas en algo más que "confirmar que funcionan". De hecho, como se comentó anteriormente, hay una serie de objetivos de las pruebas que podríamos tener, y la construcción de la confianza en el sistema es sólo uno de ellos.

Otras razones es que los probadores profesionales son más efectivos en las actividades de las pruebas. Eso tiene sentido, ¿cierto? Usted trabaja para llegar a ser bueno en algo, usted llega a ser bueno en esto, ¿no?

Esto es especialmente cierto en cuanto a encontrar fallas. No sólo los probadores tienen

habilidades en estas áreas, sino que también son más objetivos y no tienen la parcialidad del autor. Después de todo, los defectos más difíciles de encontrar serán aquellos que salgan de las suposiciones incorrectas, y el autor de algo tendrá las mismas suposiciones—correctas e incorrectas—que él formuló en sus pruebas. Una mezcla de suposiciones del probador podría consistir en una proporción igual de suposiciones correctas e incorrectas, pero por lo menos es probable que él tenga un conjunto diferente de suposiciones e improbable de compartir todas las mismas suposiciones incorrectas como las del autor.

Así que si bien es importante reiterar que los programadores pueden y deberían probar su propio código, deberíamos reconocer a ese profesional, los probadores independientes son generalmente mejores en las pruebas que los programadores, particularmente cuando un objetivo clave de las pruebas es encontrar defectos (lo cual los probadores lo hacen por medio del hallazgo de las fallas).

Ahora, como la mayoría de otras cosas en el mundo, la independencia del probador no es una situación de negro-o-blanco. Hay niveles de independencia del probador.

En el nivel más bajo de la independencia, tenemos las pruebas por medio del autor del ítem de pruebas. Esto puede ser algo efectivo, con la condición de que el autor cultive la correcta perspectiva mental de las pruebas. Las habilidades del probador faltarán normalmente.

Podemos dejar que otra persona o grupo de personas del mismo equipo haga las pruebas. Aquí tendemos a perder algunas de las cuestiones de parcialidad del autor, pero todavía estamos sujetos al pensamiento de grupo. Adicionalmente, las dinámicas de la organización—de lo cual nos ocuparemos más en el capítulo 5 acerca de la gestión de pruebas—pueden introducir presiones sobre la gente para que no critiquen el trabajo de sus colegas o causar dificultades en el equipo. Además, estamos hablando usualmente de un probador aficionado, así que otra vez las habilidades de las pruebas estarán faltando frecuentemente.

Podemos tener las pruebas por una persona o personas de un equipo diferente o por especialistas de pruebas. Ahora, probablemente podemos ver alguna reducción del problema de la brecha de las habilidades en las pruebas. Dependiendo de la estructura organizacional, se pueden reducir algunas de las presiones hacia la conformidad y lejos del desacuerdo. Empezamos a ver, en esta situación, un riesgo de la desconexión de la comunicación y el desalineamiento de los objetivos entre el equipo de pruebas y el equipo de desarrollo, el cual debe ser gestionado cuidadosamente.

En el más alto nivel de independencia, tenemos las pruebas por medio de una persona o personas de una organización o empresa diferente. Podemos tener ahora probadores con habilidades, profesionales con la plena libertad para dar una evaluación honesta sin temor a reacciones adversas.

Sin embargo, los problemas de comunicación y de los objetivos mencionados anteriormente podrían llegar a ser agudos sin una gestión cuidadosa.

Como puede ver, a lo largo de este espectro hay varios pros y contras. No es que una solución sea correcta y la otra equivocada, sino que las fortalezas y debilidades de cada método deben ser gestionadas y reforzadas. Una forma de hacerlo es garantizando las pruebas por una variedad de Participantes, con diversos grados de independencia, en todo el ciclo de vida.

Uno de los retos clave psicológicos y sociales, el cual es especialmente urgente para el jefe de pruebas, es la necesidad de tener objetivos de pruebas claros.

Como hemos mencionado en una sección anterior y nuevamente más antes en esta sección y habiendo expuesto con claridad, los objetivos bien alineados para las pruebas es crítico. Esto es así porque la gente y los proyectos son usualmente dirigidos por objetivos.

No es como cuando las personas se reúnen un día y dicen: “¡Oigan!, yo sé, ¡hagamos un proyecto!” Y otro responde: “Sí, claro, ¿Qué pasa si también tenemos una gran presión de tiempo y horas extras pagadas al final?” Y otro aún respondiendo, “¡Oh, sí, eso sería **genial!**”

No, los proyectos son emprendidos por motivos. Las metas y los objetivos del proyecto existen, aunque a veces estos no están claramente expresados. En esos casos, estos no podrían ser compartidos a través de los participantes del proyecto.

En la búsqueda de beneficios económicos y sociales, las personas usualmente tratan de alinearse usualmente con los objetivos indicados o implícitos, en la medida en que ellas entienden aquellos objetivos establecidos por la gerencia y otros interesados del negocio.

En el área de las pruebas, a menudo existe confusión acerca de los objetivos de las pruebas. Muy pocas veces están claramente definidos y bien alineados a través de toda la organización.

Al hacer evaluaciones, constantemente nos encontramos ante la confusión acerca de en qué medida las pruebas encuentran defectos o dan confianza en la operación correcta del software.

Incluso cuando estos objetivos están definidos, generalmente no hay métricas para el éxito. Es decir, si las pruebas deberían encontrar defectos, ¿Qué porcentaje de los defectos deberían encontrar? Si las pruebas son para dar confianza, ¿A qué nivel y cómo debería ser medido eso?

Por esta razón, a menudo recomendamos una política de pruebas. Una política de pruebas es un documento corto, un documento de dos a tres páginas que puede ayudar a definir claramente los

objetivos de las pruebas y alinear las pruebas a través de toda la organización. Hemos trabajado con una serie de clientes para ayudarles a poner en práctica políticas de pruebas, y esas políticas les fueron de mucha ayuda.



Figura 1.13: Enfoque

Un desafío psicológico por igual para los probadores y los jefes de pruebas es la capacidad de enfocar la atención en las cosas correctas en el momento correcto. Hay dos clases de problemas de enfoque:

La distracción de las tareas clave. Por ejemplo, el no prestar mucha atención mientras se está ejecutando un guión de pruebas y no darse cuenta de una cantidad excesiva de acceso al disco.

La persecución de las cuestiones con una mente estrecha, entonces se pierden de vista las prioridades más importantes. Por ejemplo, el comienzo de una ejecución de una prueba en un área poco probable que contenga defectos serios justo después de encontrar un defecto mayor en un área diferente del sistema, en lugar de elegir una prueba en un área del sistema relacionada con la falla recién observada.

Es importante que los probadores mantengan la absoluta concentración en una prueba mientras se está ejecutando, pero que también equilibren y reevalúen las prioridades de vez en cuando. Nosotros animamos a los probadores a preguntarse a sí mismos, al concluir una prueba, “¿Cómo influye lo que aprendí de la ejecución de esta prueba en la selección de la prueba que debería ejecutar la próxima vez?”

Para responder a esta pregunta apropiadamente, el probador debe permanecer concentrado en las metas y los objetivos del proyecto de pruebas. Para comprender aquellas metas y objetivos, el probador cuenta con la dirección clara del jefe de pruebas.



Figura 1.14: ¿Constructor o Destructor?

¿Es el probador un constructor o destructor? Él es un programador poco común quien verá al probador de la misma manera como a Vishnu, el Dios Hindú de la destrucción, pero no es nada inusual para los probadores que los hagan responsables de los daños que ellos señalaron, como si en primer lugar ellos crearan el daño.

Algunos probadores que encuentran fallas son percibidos como si criticaran al producto o el autor y también se da el caso que algunos probadores se comunican de una manera que agrava esto. Es de vital importancia para los probadores en comunicarse de manera constructiva y positiva.

A veces, las pruebas son vistas como una actividad destructiva, sin embargo, realizadas correctamente, las pruebas son esenciales para la gestión de riesgos del producto. No es que los probadores pusieran los defectos en el producto, simplemente sacan a la luz la información importante acerca de la presencia de los defectos.

Una importante regla psicológica y social es que los equipos de proyecto deben reconocer la diferencia entre hacer un daño, señalar un daño y eliminar un daño. Debería estar claro quién es responsable de cada tarea, y también debería quedar claro para todo el equipo que los probadores están allí para ayudar a alcanzar el mejor resultado posible.



Figura 1.15: ¿Malas Noticias o un Mal Individuo?

Este desafío psicológico del constructor/destructor proviene de un fenómeno psicológico fundamental en la mente humana que es llamado "transferencia". La gente tiende a transferir al mensajero cómo se siente acerca del mensaje, esto significa que algunas veces los probadores están recibiendo el final de las emociones traídas por las noticias de los problemas del proyecto.

Usted no puede cambiar la naturaleza humana, pero buenas habilidades de comunicación pueden ayudar.

Por un lado, enfatizar que su objetivo como un probador profesional es de colaborar con el equipo

para una mejor calidad, no hablar constantemente acerca de los problemas, no hacer que la gente se sienta estúpida o no tener razón acerca de lo deficiente que es el producto.

Comuníquese neutralmente acerca de los hechos, sin crítica del producto o de la gente involucrada. Evitando que la gente se ponga a la defensiva, porque ellos dejarán de escucharle si eso ocurre. Eso significa que usted sería inefectivo en su rol.

Comprender a sus colegas y observar cómo reaccionan a sus hallazgos. ¿Cuál es la mejor manera de comunicar malas noticias a cada persona con la que trabaja? Adapte su mensaje para una entrega efectiva.

Asegúrese que usted es claro y confirme que sus colegas entendieron lo que dijo. Confirme que usted entiende a sus colegas. La comunicación deficiente ocurre. Las comunicaciones deficientes no son divertidas.

También es importante tomar en cuenta que la presión de los cronogramas puede agravar este problema de transferencia. Esto significa que, durante el período de las típicas pruebas de sistema en el final del proyecto, la tendencia humana de reaccionar negativamente a las malas noticias estará en su peor momento. Esa es necesariamente una noticia reconfortante, pero por lo menos podemos tratar de comportarnos de acuerdo a esa situación.

1.5.1 Ejercicios

Ejercicio 1

Reflexione acerca de las actitudes y los comportamientos de los probadores más exitosos que conoce.

¿En qué medida muestran ellos los aspectos psicológicos abordados en esta sección?

¿Qué otros elementos de sus conjuntos de personalidades y habilidades piensa usted que conduce al éxito para ellos?

Argumente.

Solución del Ejercicio 1

La curiosidad, la experiencia, el pesimismo profesional, el enfoque, una perspectiva crítica e interrogativa, la atención al detalle y las buenas habilidades de comunicación son todos esenciales para los probadores. Es difícil imaginar un probador exitoso que no exhiba todos o al menos la mayoría de estos rasgos. De hecho, hemos trabajado con probadores que tenían todos los rasgos excepto uno—en un caso el enfoque, en otro caso, el pesimismo profesional—y ese rasgo faltante limitaba seriamente su potencial.

Las habilidades de comunicación se vuelven especialmente importantes a medida que usted se mueve hacia arriba a las posiciones más séniores de líder de pruebas o de dirección.

No es sólo la falta de estos rasgos importantes que pueden causar problemas. Hay algunas personas a quienes consideraríamos como no exitosos quienes exhiben todos estos rasgos. Sin embargo, ellas tienen uno o dos rasgos adversos, así como la arrogancia, las inseguridades compensadas excesivamente y la actitud defensiva, lo cual les causa problemas repetidos con los compañeros y los colegas. Las pruebas son un trabajo en equipo, y usted tiene que aprender a trabajar adecuadamente con los miembros del equipo como un compañero y como un líder para ser un probador exitoso.

1.6 Código de Éticas

Objetivos del Aprendizaje

No hay objetivos del aprendizaje definidos para el capítulo 1, sección 6 del programa de estudios (syllabus).

Esta sección, Código de Éticas, cubrirá los siguientes conceptos clave:

- Comprensión de un código de éticas profesional.
- Aplicación de un código de éticas profesional para situaciones determinadas.

Muchas profesiones tienen estándares éticos. En el contexto del profesionalismo, las éticas son “reglas de conducta reconocidas con respecto a una clase particular de acciones humanas o a un grupo particular, una cultura, y etc.”.

Porque los probadores tienen acceso a información confidencial y privilegiada, las guías éticas pueden ayudar a guiarlos a utilizar esa información correctamente. Adicionalmente, los probadores deberían utilizar guías éticas para seleccionar los mejores comportamientos y resultados posibles para una situación dada, teniendo en cuenta sus restricciones. Note que “Lo mejor posible” significa para cada uno, no sólo para el probador.

Permítanos presentarle un ejemplo de éticas en acción. Rex Black es un gerente de tres consultoras de pruebas de software internacionales que están relacionadas, RBCS, RBCS NZ y Software Test Worx. También él tiene cargos en las directivas de los comités del ISTQB y ASTQB.

Como tal, él podría tener y tiene la percepción de la dirección del programa que sus competidores no lo tienen en el negocio de consultoría en pruebas de software.

En algunos casos, así como el caso de su ayuda para desarrollar el programa de estudios (syllabus), él tiene que hacer claro esos intereses de negocios a las personas, pero también le es permitido ayudar de esa manera. Ayudó a escribir ambos, el programa de estudios (syllabus) Básico y Avanzado.

En otros casos, así como en el desarrollo de las preguntas de examen, él acordó, junto con sus colegas en el ASTQB, que no debería participar en eso. El acceso directo a las preguntas de examen, haría demasiado probable, que él conscientemente o inconscientemente, tergiversaría sus materiales de capacitación para "enseñar el examen".

En la medida que avance en su carrera como un probador, más y más oportunidades de mostrar su naturaleza ética—o de ser engañado por la falta de ésta—se aparecerán en su camino. Nunca es demasiado temprano para inculcar un sentido fuerte de éticas.

En el nivel Básico, el ISTQB espera que los que son certificados se adhieran al siguiente código de éticas con respecto a:

EL PÚBLICO - Los probadores de software certificados deberían actuar coherentemente con el interés del público. Por ejemplo, si está trabajando en un sistema de seguridad crítica y se le pide silenciosamente que cancele algunos informes de los defectos, ése es un problema ético.

EL CLIENTE Y EMPLEADOR - Los probadores de software certificados deberían actuar de una manera que es en el mejor interés de su cliente y empleador, y de conformidad con el interés público. Por ejemplo, si sabe que el proyecto principal de su empleador está con problemas, y usted vende la acción a bajo precio entonces se escapa información acerca de los problemas del proyecto a Internet, esto es una real falta ética—y también probablemente un delito.

EL PRODUCTO - Los probadores deberían asegurar que los entregables que ellos proporcionan (acerca de los productos y los sistemas que ellos prueban), cumplan con los más altos estándares posibles. Por ejemplo, si está trabajando como un consultor y deja afuera detalles importantes de un plan de pruebas de tal forma que el cliente tiene que contratarlo en el próximo proyecto, eso es una falta ética.

EL JUICIO - Los probadores de software certificados deberían mantener la integridad e independencia en su juicio profesional. Por ejemplo, si un jefe de proyecto le pide no informar acerca de los defectos en ciertas áreas por temor a las reacciones potenciales de los inversores del negocio, eso es un golpe a su independencia y un fracaso ético de su parte si usted accede a su pedido.

LA GESTIÓN - Los jefes y líderes de pruebas de software certificados suscribirán y promoverán un método ético para la gestión de las pruebas de software. Por ejemplo, si usted favorece a un probador más que a otro, porque le gustaría establecer una relación romántica con la hermana del probador favorecido es una falta seria de éticas de gestión.

LA PROFESIÓN - Los probadores de software certificados deberían promover la integridad y reputación de la profesión en conformidad con el interés público. Por ejemplo, si tiene una oportunidad de explicar a los compañeros de su hijo o a los colegas de su esposa lo que hace, esté orgulloso de ello y explique cómo las pruebas de software benefician a la sociedad.

LOS COLEGAS - Los probadores de software certificados deberían ser justos con sus colegas además de ser de gran ayuda para ellos y promover la cooperación con los desarrolladores de software. Por ejemplo, no es ético manipular los resultados de las pruebas para acordar el despido de un programador a quién usted deteste.

UNO MISMO - Los probadores de software certificados deberían participar en el aprendizaje permanente con relación a la práctica de su profesión y deberían promover un método ético para la práctica de la profesión. Por ejemplo, la profesión, la participación en cursos, la lectura de libros y la exposición en conferencias acerca de lo que usted hace le ayuda a avanzar. Esto se denomina "me está yendo bien mientras lo estoy haciendo bien", y afortunadamente, ¡es muy ético!

Glosario del ISTQB

Revisión: Una evaluación de un producto o estado de un proyecto para determinar las discrepancias desde los resultados planificados hasta las mejoras recomendadas. Por ejemplo la revisión de gestión, la revisión informal, la revisión técnica, la inspección y la revisión guiada.

Preguntas de Examen de Muestra y Simulación

Para finalizar cada capítulo, usted puede tratar de resolver una o más preguntas de examen de muestra para reforzar su conocimiento y comprensión del material y prepararse para el examen del Probador ISTQB Nivel Básico.

Sección 1.1: ¿Por qué son necesarias las pruebas? (K2)

#	Pregunta	
1.	<p>Objetivo del aprendizaje: LO-1.1.1</p> <p>¿Cuál de los siguientes escenarios describen una forma en la cual un defecto del software puede causar principalmente y directamente daños a una compañía?</p> <ul style="list-style-type: none"> A. Un banco que genera el 5% de sus ingresos de las comisiones del cajero automático sufre una caída de su red debido a un defecto del software. B. El software de navegación de un automóvil muestra rutas a través de ríos utilizando el mismo ícono tanto para una balsa como un puente. C. El software de monitoreo de emisiones informa demasiado poco acerca del porcentaje de gases tóxicos en una fábrica. D. Un encargado de ventas escribe la letra "l" donde el numeral "1" debería haber ido, por lo que una propuesta importante no fue entregada a un cliente potencial. 	2
2.	<p>Objetivo del aprendizaje: LO-1.1.2</p> <p>Considerar la siguiente secuencia de eventos.</p> <ol style="list-style-type: none"> I. Un jefe decide eliminar las revisiones del código en un intento de acelerar las pruebas. II. Una operación esencial de datos, de la cual otro subsistema depende en una aplicación bancaria, es eliminada por un programador que está realizando el trabajo de mantenimiento en un fragmento de código desconocido III. Un cajero automático no puede actualizar la dirección de un cliente después de que la entrega de mantenimiento es instalada. IV. El cliente impactado por el software de mala calidad que está siendo utilizado para administrar su dinero, se cambia de banco. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. I es el defecto, II es la causa raíz, y III y IV son efectos. B. I es el efecto, II es la causa raíz, y III y IV son defectos. C. I es la causa raíz, II y III son defectos y IV es el efecto. D. I es la causa raíz, II es el defecto, y III y IV son efectos. 	2
3.	<p>Objetivo del aprendizaje: LO-1.1.3</p> <p>¿Cuál de los siguientes es el mejor ejemplo de por qué las pruebas son necesarias?</p> <ul style="list-style-type: none"> A. Los Jefes de proyectos escriben los planes del proyecto. Lo interesados del proyecto no siempre revisan los planes de proyecto. B. El software moderno puede contener más de un millón de líneas de código. Según estudios de la industria, el programador promedio de C, introduce un defecto por cada 25 líneas de código. C. Los usuarios tienen ideas inusuales acerca de lo que el software debería poder hacer. Los programadores no entienden a los usuarios. D. El personal del soporte técnico confía en soluciones alternativas para asistir a los clientes con defectos conocidos. Los probadores son las únicas personas que encuentran soluciones alternativas para los defectos. 	2
4.	<p>Objetivo del aprendizaje: LO-1.1.4</p> <p>¿Cuál de los siguientes es un ejemplo de que las pruebas contribuyen a mejorar la calidad?</p> <ul style="list-style-type: none"> A. Un Jefe de proyecto le pide a un líder de pruebas que estime el esfuerzo de las pruebas. B. Un probador instala un ítem de pruebas en el entorno de pruebas. C. Un probador encuentra un defecto que es resuelto antes de la liberación. D. Un líder de pruebas escribe un informe del resumen de las pruebas. 	2
5.	<p>Objetivo del aprendizaje: LO-1.1.5</p> <p>¿Hay la misma relación entre el significado de las palabras "equivocación" y "error" que entre los siguientes pares de palabras?</p> <ul style="list-style-type: none"> A. "Falla" y "bug". B. "Falla" y "defecto". C. "Error" y "defecto". D. "Defecto" y "bug". 	1
6.	<p>Objetivo del aprendizaje: término</p> <p>Un componente de calidad es el que:</p> <ul style="list-style-type: none"> A. Satisface las necesidades del usuario y cliente. B. Fue terminado a tiempo C. Requirió menos esfuerzo que la cantidad estimada 	1

- D. Estuvo sujeto a una revisión de código.

Sección 1.2: ¿Qué son las pruebas? (K2)

#	Pregunta	K
7.	<p>Objetivo del aprendizaje: LO-1.2.1 Considerar los siguientes objetivos.</p> <ul style="list-style-type: none"> I. Encontrar los defectos. II. Ganar la confianza en el sistema. III. Proporcionar información acerca del sistema. IV. Prevenir los defectos. V. Eliminar los defectos. <p>¿Cuál de los siguientes enumeran los objetivos comunes de las pruebas?</p> <ul style="list-style-type: none"> A. I, II, III, IV, y V. B. I, II, y IV. C. I, II, III, y IV. D. I y V. 	1
8.	<p>Objetivo del aprendizaje: LO-1.2.2 Un probador participa en una revisión de requisitos e identifica ambigüedades que podrían haber resultado en equivocaciones de programación. Éste es un ejemplo de</p> <p>¿Cuál de los siguientes propósitos de las pruebas?</p> <ul style="list-style-type: none"> A. Encontrar defectos. B. Proporcionar confianza. C. Prevenir defectos. D. Proporcionar información. 	2
9.	<p>Objetivo del Aprendizaje: LO-1.2.3 ¿Cuál de los siguientes es un ejemplo de la depuración?</p> <ul style="list-style-type: none"> A. La observación de una anomalía. B. La repetición de la prueba de una corrección del defecto. C. La preselección de un defecto. D. La Corrección de un defecto. 	2
10.	<p>Objetivo del aprendizaje: término. ¿Cuáles actividades están involucradas en las pruebas?</p> <ul style="list-style-type: none"> A. Las actividades al final del ciclo de vida del software. B. Sólo las actividades que evalúan los productos del software. C. Las actividades que involucran la ejecución de una prueba por el componente o el sistema sometido a pruebas. D. Las actividades a través de todo el ciclo de vida del software, incluyendo la planificación, preparación y evaluación. 	1

Sección 1.3: Principios generales de las pruebas (K2)

#	Pregunta	K
11.	<p>Objetivo del Aprendizaje: LO-1.3.1 Considerar el siguiente escenario. Usted está probando un producto y ha encontrado 100 defectos. Dos tercios de esos defectos están en la interfaz de usuario y los módulos del control de acceso, mientras que los restantes 33 están esparcidos entre los otros 6 módulos. ¿Cuál de los siguientes es el principio que más aplica a este escenario?</p> <ul style="list-style-type: none"> A. El agrupamiento de defectos. B. La falacia de la ausencia de errores. C. Las pruebas exhaustivas son imposibles. D. Controlar activamente el diseño de las pruebas mientras realiza esas pruebas. 	2
12.	<p>Objetivo del aprendizaje: término. Las pruebas exhaustivas involucran:</p> <ul style="list-style-type: none"> A. Todos los pares de los valores de entrada y las precondiciones 	1

- B. Por lo menos uno de cada valor posible de las entradas y las precondiciones.
- C. Todas las combinaciones de los valores de entrada y las precondiciones.
- D. Todas las combinaciones de los valores de entrada y salida.

Sección 1.4: Proceso de pruebas básico (K1)

#	Pregunta	K
13.	<p>Objetivo del aprendizaje: LO-1.4.1</p> <p>¿Cuál de las siguientes tareas de las pruebas es una parte de la actividad de la planificación de las pruebas?</p> <ul style="list-style-type: none"> A. Medir y analizar los resultados. B. Determinar los criterios de salida. C. Revisar las bases de las pruebas. D. Comprobar los registros de las pruebas contra los criterios de salida. 	1
14.	<p>Objetivo del aprendizaje: término</p> <p>¿Por qué ejecutamos las pruebas de confirmación?</p> <ul style="list-style-type: none"> A. Para demostrar la aptitud para el propósito. B. Para asegurar que los defectos no han sido introducidos en las áreas no modificadas. C. Para determinar si un componente o sistema satisface o no las necesidades de un usuario/cliente. D. Para verificar el éxito de las acciones correctivas 	1

Sección 1.5: La psicología de las pruebas (K2)

#	Pregunta	K
15.	<p>Objetivo del aprendizaje: LO-1.5.1</p> <p>¿Por qué es importante declarar claramente los objetivos de pruebas?</p> <ul style="list-style-type: none"> A. Porque la gente tiende a alinear sus planes con los objetivos establecidos por la gerencia. B. Porque la cobertura debe ser medida contra los objetivos de las pruebas. C. Porque el nivel de riesgo del producto es determinado por los objetivos de las pruebas. D. Porque la identificación de los objetivos de las pruebas es una tarea principal en las actividades de planificación. 	1
16.	<p>Objetivo del aprendizaje: LO-1.5.2</p> <p>Para mantener la motivación, los probadores y desarrolladores necesitan por igual la mentalidad correcta. Los buenos desarrolladores deben tener una actitud positiva acerca de su habilidad y la habilidad de su equipo para tratar los riesgos de negocios y técnicos que confrontan sus proyectos de programación. ¿Cuál de las siguientes describe exactamente la mentalidad opuesta de un buen probador?</p> <ul style="list-style-type: none"> A. Los buenos probadores identifican fallas con la intención de ser críticos acerca del producto y el autor. B. Los buenos probadores dicen la verdad como la ven y no están terriblemente preocupados en cómo la gente va a reaccionar a esos hechos. C. Los buenos probadores son profesionales pesimistas que creen que el producto contiene probablemente defectos y que ellos pueden encontrarlos. D. Los buenos probadores se enfocan más en los riesgos técnicos que en los riesgos de negocios, porque ellos quieren encontrar tantos defectos como sea posible. 	2
17.	<p>Objetivo del aprendizaje: término</p> <p>¿Cuál meta fomenta la independencia de las pruebas?</p> <ul style="list-style-type: none"> A. La reducción de los costos de las pruebas antes de la liberación. B. La maximización de las pruebas por medio de aquellos que están familiarizados con el código. C. El mejoramiento de la comunicación y las relaciones entre los probadores y otros. D. Proporcionar un mayor grado de objetividad y reducir la parcialidad del autor. 	1

Sección 1.6: Código de éticas (K2)

#	Pregunta	K
18.	<p>Objetivo del Aprendizaje No disponible.</p> <p>Un probador está trabajando en un proyecto importante para su compañía, la cual es grande y cotiza en la bolsa. El encuentra muchos defectos durante las pruebas, lo cual él cree que provocará que el proyecto fracase, con un efecto negativo en los precios de las acciones de la compañía. Él comparte esta información con su primo, quién utiliza dicha información para colocar negocios en contra de la compañía en el mercado de acciones. Éste es un ejemplo de falta de ética relacionada a ¿Cuál de los siguientes?</p> <ul style="list-style-type: none"> A. El cliente y empleador. B. El público. C. El juicio. D. La gestión. 	2

Capítulo 1 Pregunta que cubre secciones

#	Pregunta	K
19.	<p>Cubre: Sección 1.1, 1.2, y 1.4.</p> <p>Un objetivo común de las pruebas es el suministro de información. Las pruebas deben proporcionar la suficiente información a los interesados del negocio para tomar decisiones informadas acerca de la versión del software o sistema que está siendo probado. ¿Cuál de las siguientes es una actividad fundamental del proceso de pruebas, durante la cual es evaluada la cantidad suficiente de las pruebas y la información resultante?</p> <ul style="list-style-type: none"> A. El análisis y diseño. B. La implementación y ejecución. C. La evaluación del criterio de salida y la creación de informes. D. La especificación de requisitos. 	2

Preguntas del Examen de simulación 1

#	Pregunta	K
20.	<p>Objetivo del aprendizaje: LO-1.1.4 Un jefe de pruebas presenta un informe del resumen del estado de las pruebas al equipo del proyecto. Basado en este informe, el equipo del proyecto decide extender el período de la ejecución de las pruebas por tres semanas para permitir que defectos adicionales sean encontrados y corregidos. ¿Qué ilustra esta situación?</p> <ul style="list-style-type: none"> A. La contribución de las pruebas a la más alta calidad. B. El proceso básico de pruebas. C. La mentalidad opuesta de los desarrolladores y probadores. D. La falacia de la ausencia de errores. 	2
21.	<p>Objetivo del aprendizaje: LO-1.2.1 Considerar lo siguiente:</p> <ul style="list-style-type: none"> I. Prevenir los defectos. II. Eliminar los defectos. III. Ganar la confianza en el sistema. IV. Proporcionar información. V. Encontrar los defectos. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Todos son objetivos comunes de las pruebas. B. I, II, III, y V son objetivos comunes de las pruebas. C. I, III, IV, y V son objetivos comunes de las pruebas. D. II y V son objetivos comunes de las pruebas. 	1
22.	<p>Objetivos del aprendizaje: 1) LO-1.3.1 y 2) LO-1.4.1</p> <p>¿Cuál de los siguientes es un principio general de las pruebas?</p> <ul style="list-style-type: none"> A. La planificación y el control. B. Las pruebas tempranas. C. El análisis y diseño. D. La implementación y ejecución. 	1

<p>23. Objetivo del aprendizaje: LO-1.5.2 Un probador está realizando una prueba. El observa una inusual cantidad de accesos a la red durante un período, cuando él sabe que la aplicación no debería acceder a la red. Aún cuando la prueba no especifica la comparación del acceso real a la red con un resultado esperado, el probador decide investigar. Éste es un ejemplo ¿de qué?</p> <ul style="list-style-type: none"> A. Una mentalidad curiosa del probador. B. Una comunicación efectiva del probador con el desarrollo. C. Un probador realizando pruebas exhaustivas. D. Un probador realizando las pruebas de mantenimiento. 	2
<p>24. Objetivo del aprendizaje: término ¿Qué son las pruebas de regresión? Las pruebas que ejecutan los casos de prueba que fallaron la última vez que fueron ejecutados.</p> <ul style="list-style-type: none"> B. Las pruebas de un programa ya antes probado tras a una modificación. C. Un sinónimo de pruebas de confirmación. D. Las pruebas para determinar la fiabilidad de un producto de software. 	1
<p>25. Objetivo del aprendizaje: LO-1.3.1 Está ejecutando una prueba escrita que otros probadores la han ejecutado previamente. Adicionalmente esta prueba ha encontrado defectos previamente, y esos defectos han sido confirmados para ser corregidos. El jefe de pruebas le recomienda a usted a variar la forma específica en la cual usted ejecuta las pruebas, así como el orden de ciertas acciones, la utilización del ratón versus las teclas de acceso rápido y los valores de entrada específicos, basados en la manera en que los usuarios utilizarán el sistema. ¿Cuál de los siguientes es un principio de las pruebas que podría explicar la indicación del jefe de pruebas?</p> <ul style="list-style-type: none"> A. Las pruebas tempranas. B. Las pruebas aleatorias. C. La paradoja del pesticida. D. La falacia de la ausencia de errores. 	2
<p>26. Objetivos del aprendizaje: 1) LO-1.1.4 y 2) LO-1.3.1 Las pruebas pueden:</p> <ul style="list-style-type: none"> A. Medir la calidad del software según los defectos encontrados B. Eliminar la posibilidad de que no hay defectos no descubiertos. C. Encontrar nuevos defectos a través de la repetición de las mismas pruebas. D. Encontrar, analizar y eliminar las causas de las fallas. 	1

Preguntas del Examen de Simulación 2

#	Pregunta	K
27.	<p>Objetivo del aprendizaje: término ¿Qué es cierto de un sistema de calidad?</p> <ul style="list-style-type: none"> A. Es seguro. B. Satisface los requisitos especificados, las necesidades y las expectativas del cliente y los usuarios. C. Cuesta más que otros sistemas con las mismas características. D. Fue entregado a tiempo y dentro del presupuesto. 	1
28.	<p>Objetivo del aprendizaje: LO-1.1.2 Un teléfono que suena momentáneamente en un cubículo de al lado distrae a un programador, causando que él programe incorrectamente la lógica que verifica el límite superior de una variable de entrada. Después, durante las pruebas de sistema, un probador observa que esta variable acepta valores de entrada inválidos. La lógica codificada incorrectamente para el límite superior es:</p> <ul style="list-style-type: none"> A. La causa raíz. B. La falla. C. Un ejemplo de enmascaramiento de faltas. D. El defecto. 	2
29.	<p>Objetivo del aprendizaje: LO-1.2.1 ¿Cuál de los siguientes es un objetivo común de las pruebas?</p>	1

	<p>A. Prevenir los defectos. B. Eliminar los defectos. C. Comparar los resultados reales con los resultados esperados. D. Analizar la causa de la falla.</p>	
30.	<p>Objetivo del aprendizaje: LO-1.3.1 ¿Cuál de los siguientes es un principio general de las pruebas?</p> <p>A. Las pruebas exploratorias. B. El agrupamiento de defectos. C. La evaluación de los criterios de salida. D. El enmascaramiento de defectos.</p>	1
31.	<p>Objetivo del aprendizaje: LO-1.4.1 ¿Cuál de las siguientes es una actividad básica de las pruebas?</p> <p>A. La depuración. B. La implementación y ejecución de las pruebas. C. La construcción diaria. D. Las pruebas de confirmación.</p>	1
32.	<p>Objetivo del aprendizaje: LO-1.5.2 Un programador está escribiendo y ejecutando las pruebas unitarias contra el código que ha escrito. ¿Cuál de las siguientes es una mentalidad de probador que el programador debería adoptar para realizar estas pruebas unitarias efectivamente?</p> <p>A. Las buenas habilidades de comunicación. B. La cobertura de código. C. El efecto sonda. D. La atención al detalle.</p>	2
33.	<p>Objetivos del aprendizaje: 1) LO-1.1.3 y 2) LO-1.3.1 Un programador está trabajando en un código muy complejo. ¿Cuál de los siguientes es un principio general de las pruebas que puede afectar este trabajo?</p> <p>A. El agrupamiento de defectos. B. La paradoja del pesticida. C. Las pruebas tempranas. D. Las pruebas de componente.</p>	2

6 LO es una abreviación de la palabra en inglés Learning Objective y K es Knowledge respectivamente. Estas dos abreviaciones hacen referencia a los objetivos del aprendizaje y al nivel de conocimiento de cada objetivo como está especificado en el Programa de Estudios Nivel Básico 2011. (Syllabus 2011). Para mayor información visitar www.istqb.org

7 Este ejercicio y su solución han sido adaptados del capítulo 2, del libro de Rex Black, *Pragmatic Software Testing*.

8 <http://es.wikipedia.org>: Se usa como insecticida en cosechas agrícolas y en jardines, para tratar piojos en la cabeza de seres humanos y para tratar pulgas en animales domésticos.

9 es.wikipedia.org: En el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

10 Aunque no está definido en la Real Academia Española es un término muy utilizado en la informática para indicar la capacidad de algo que pueda ser probado o comprobado.

11 Autoprueba no está definido en la Real Academia Española, pero es un término utilizado en el ámbito de las pruebas de software y se define como la prueba que es realizada por el mismo autor o creador del objeto que debe ser probado.

Capítulo 2

Pruebas a través del Ciclo de Vida de Software

"En el pensamiento científico siempre están presentes elementos de poesía. La ciencia y la música actual exigen de un proceso de pensamiento homogéneo"

Albert Einstein, premio nobel de física de la teoría de la relatividad.

El capítulo 2, Pruebas a través el Ciclo de Vida de Software, contiene las siguientes cuatro secciones:

1. Modelos de desarrollo de software.
2. Niveles o fases de pruebas.
3. Tipos de pruebas.
4. Pruebas de mantenimiento.

La mayoría de las secciones estarán desglosadas en dos o más partes.

2.1 Modelos de Desarrollo de Software

Objetivos del Aprendizaje

LO-2.1.1 Explicar la relación entre el desarrollo, las actividades de pruebas y los productos del trabajo en el ciclo de vida del desarrollo por medio de ejemplos utilizando los tipos de proyecto y producto. (K2)

LO-2.1.2 Reconocer el hecho de que los modelos de desarrollo de software deben ser adaptados al contexto del proyecto y a las características del producto. (K1)

LO-2.1.3 Recordar las características de las buenas pruebas, que son aplicables en cualquier modelo del ciclo de vida. (K1)

Esta sección, Modelos de Desarrollo de Software, cubrirá los siguientes conceptos clave:

- La relación entre las actividades de desarrollo y pruebas.
- Cómo adaptar los modelos de desarrollo de software al contexto del proyecto y producto.
- Características de las buenas pruebas a través de todos los modelos.

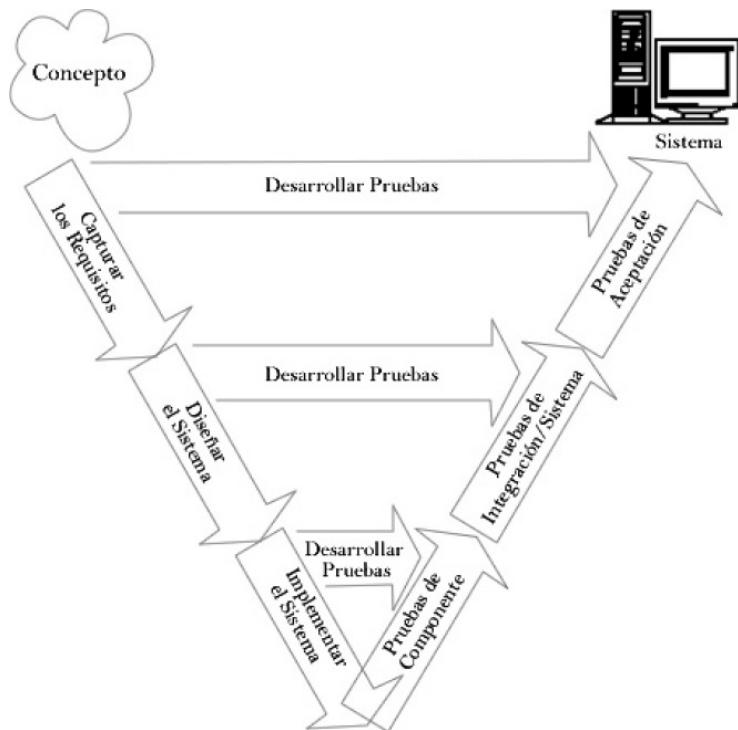


Figura 2.1: Modelo V o Secuencial

En esta figura se puede observar una representación del modelo V.

Glosario del ISTQB

Modelo V: Un framework para describir las actividades del ciclo de vida del desarrollo de software desde la especificación de requisitos hasta el mantenimiento. El modelo V ilustra cómo las actividades de pruebas pueden ser integradas en cada fase del ciclo de vida del desarrollo de software.

El modelo V y otros modelos secuenciales tienen como principal característica el refinamiento del concepto jerárquico en el sistema en una secuencia individual y lineal de actividades.

Las flechas horizontales de la V, denominadas “Desarrollar las Pruebas”, indican la planificación, el análisis, el diseño y la implementación de las pruebas. La flecha se origina de la etapa del ciclo de vida que produce una o más bases principales de las pruebas para el nivel correspondiente de pruebas. En otras palabras, las especificaciones de los requisitos son la base principal para las pruebas de aceptación. Las especificaciones de los requisitos y el diseño de alto nivel son las bases principales de las pruebas para las pruebas de sistema. El diseño detallado y de alto nivel es la base principal de las pruebas para las pruebas de integración. El diseño detallado y el código en sí son las bases principales de las pruebas para las pruebas unitarias.

Glosario del ISTQB

Requisito funcional: Un requisito que especifica una función, la cual un componente o sistema debe realizar.

Requisito no funcional: Un requisito que no está relacionado con la funcionalidad, pero sí con los atributos como fiabilidad, eficiencia, usabilidad, mantenibilidad y portabilidad.

Integración: El proceso de la combinación de los componentes o sistemas en ensamblados más grandes.

Pruebas de integración: Pruebas realizadas para exponer defectos en las interfaces y las interacciones entre los componentes o sistemas integrados. Véase también pruebas de integración de componentes, pruebas de integración de sistemas.

Pruebas de sistema: El proceso de probar un sistema integrado para verificar que cumple los requisitos especificados.

Nivel de pruebas: Un grupo de actividades de las pruebas que están organizadas y gestionadas conjuntamente. Un nivel de pruebas está conectado a las responsabilidades en un proyecto. Ejemplos de niveles de pruebas son las pruebas de componente, pruebas de integración, pruebas de sistema y pruebas de aceptación.

Pruebas de aceptación del usuario: Véase pruebas de aceptación.

Pruebas de aceptación: Pruebas formales con respecto a las necesidades de los usuarios, los requisitos y los procesos de negocio, dirigidas para determinar si un sistema satisface o no los criterios de aceptación y para permitir al usuario, los clientes u otra entidad autorizada para determinar si aceptan o no el sistema. Tenga en cuenta que este término no se citó específicamente en esta sección, pero se lo incluye aquí porque es un sinónimo de las pruebas de aceptación del usuario.

Pruebas de componente (también conocidas como pruebas de unidad, módulo o programa):

Las pruebas de componentes individuales de software.

El tiempo de la secuencia de la animación muestra la secuencia aproximada de las actividades. Así, puede ver que la planificación de pruebas, el análisis, el diseño y las tareas de implementación, mostradas en las flechas horizontales de la V, son activados por la creación de sus bases principales de las pruebas.

Éste es un modelo intuitivo y usual. La mayoría de los profesionales de TI, si han trabajado en algunos proyectos, se han encontrado con este método. Sin duda es preferible este modelo que el caos o la falta de cualquier estructura del ciclo de vida o modelo en absoluto.

Desde el punto de vista de las pruebas, éste tiene algunas imperfecciones. Por un lado, es dirigido usualmente por los riesgos de los cronogramas y los presupuestos, más que dirigido por los riesgos de calidad. Por otro lado, porque es difícil de planificar por adelantado sin olvidar nada en los proyectos más grandes cuando los planes fracasan, las pruebas—especialmente las pruebas de sistema y aceptación—¡tienen dificultades al final! Estas imperfecciones pueden ser gestionadas mediante una cuidadosa gestión de ambos, los riesgos de calidad y los riesgos de proyecto relacionados con las pruebas. Hablaremos más acerca de la gestión de riesgos en los capítulos 4 y 5.

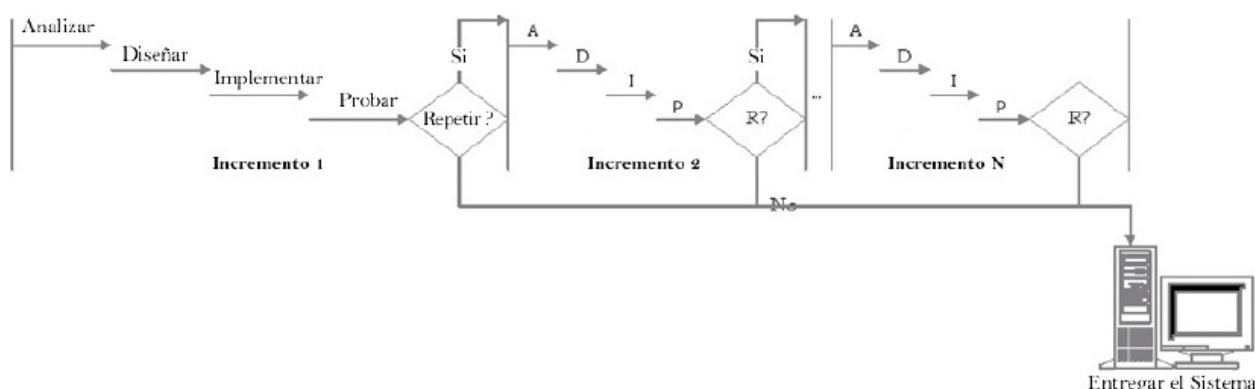


Figura 2.2: Modelo Iterativo, Evolutivo o Incremental

Los modelos iterativo, evolutivo o incremental tienen como principal característica la transformación del concepto en el sistema en una serie de secuencias de actividades repetidas y lógicas. Cada secuencia de las actividades se denomina una iteración o un incremento o un salto (“sprint”) o algún otro nombre, dependiendo del modelo.

Cada iteración, incremento, “sprint” o lo que sea, produce algunos conjuntos de características y alguna parte ejecutable más grande del sistema, lo cual podría ser por sí mismo un entregable

valioso para los usuarios.

Glosario del ISTQB

Modelo de desarrollo iterativo-incremental: Este término no está definido directamente en el glosario, pero son dos términos relacionados.

Modelo de desarrollo iterativo: Un ciclo de vida del desarrollo donde usualmente un proyecto es dividido en un gran número de iteraciones. Una iteración es un ciclo completo del desarrollo que resulta en una versión (interna o externa) de un producto ejecutable, un subconjunto del producto final en desarrollo, que crece de iteración en iteración para convertirse en el producto final.

Modelo de desarrollo incremental: Un ciclo de vida del desarrollo donde un proyecto es dividido en una serie de incrementos, cada uno de los cuales produce una parte de la funcionalidad del total de los requisitos del proyecto. Los requisitos son priorizados y entregados en orden de prioridad en el incremento adecuado. En algunas (pero no en todas) versiones de este modelo del ciclo de vida, cada subproyecto sigue un "mini modelo V" con sus propias fases de diseño, codificación y pruebas.

Considere el conjunto de las características como ése que gradualmente crece alrededor de la funcionalidad esencial, que fue desarrollado en el primer incremento. Si la funcionalidad esencial es independiente y de valor, entonces usted puede entregar algo en cualquier momento una vez que la funcionalidad esencial esté lista y pase sus pruebas.

Debido a esta característica, estos métodos tienden a ser muy útiles en situaciones dirigidas por los riesgos de cronograma. Si hay alguna flexibilidad acerca de lo que debe ser entregado exactamente, pero la necesidad de alcanzar una fecha límite es grave, este método es mejor que el método V.

Más o menos desde 1995, los modelos del tipo iterativo, que se clasifican según su formalidad desde la Programación Extrema pasando por el Desarrollo Rápido de Aplicaciones hasta el Proceso Unificado, se han vuelto bastante populares, a medida que las fechas límites inflexibles de los cronogramas se han convertido en la normalidad.

Sin embargo, no todo es perfecto desde el punto de vista de las pruebas.

Por un lado, hay una gran probabilidad de que las pruebas del incremento final puedan ser apresuradas o abreviadas bajo presión. Esto puede resultar en entregables defectuosos.

Por otro lado, si bien no es visible en la imagen de la figura 2.2, en realidad hay una buena cantidad de superposición entre los incrementos. Los programadores, una vez que completan su trabajo en un incremento, pasan al siguiente. Esto hace que sea difícil para ellos corregir los defectos encontrados durante las pruebas del anterior incremento y puede atrasar el progreso a través de las pruebas.

Además, debido a que los programadores están cambiando y añadiendo código en cada incremento sucesivo, los riesgos de regresión son muy altos, particularmente para el código entregado de alto valor en el primer incremento.

Finalmente, en los proyectos que se siguen algunos de los modelos llamados "ágiles", el rol de las pruebas está todavía evolucionando. Algunos creadores de los métodos ágiles han sido, francamente, completamente hostiles hacia el concepto de las pruebas independientes y los probadores independientes, lo cual significa que usted podría encontrarse obligado a forjar su rol.

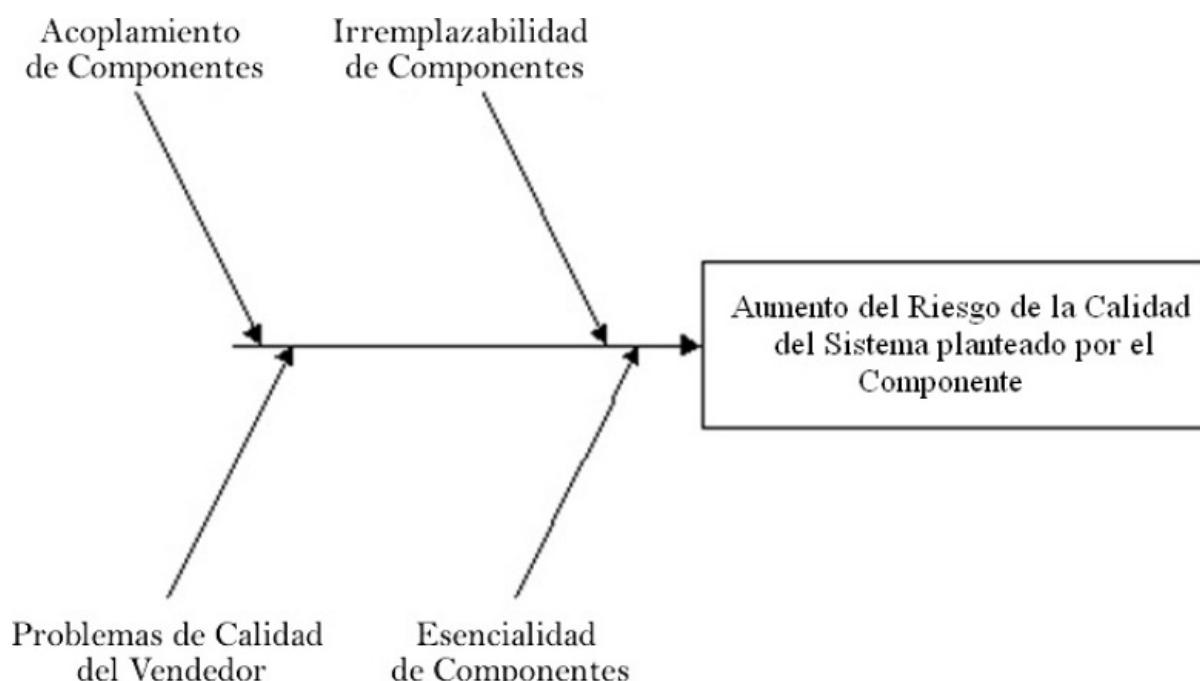


Figura 2.3: Integración de Sistemas

Los proyectos de integración se realizan cuando integramos principalmente un software existente (p.ej., un Software Comercial de Distribución Masiva), o cuando externalizamos el desarrollo de software y luego integramos esos componentes adquiridos de terceros para formar un sistema en vez de que construyamos un nuevo software.

Glosario del ISTQB

Comercial de distribución masiva (COTS): Véase software de distribución masiva.

Software de distribución masiva: un producto de software que es desarrollado para el mercado masivo, es decir para un gran número de clientes y que es entregado a muchos clientes en formato idéntico. Tenga en cuenta que este término no es citado específicamente en esta sección, pero se lo incluye aquí, porque es un sinónimo de comercial de distribución masiva.

A pesar de las nociones comunes acerca de estos métodos, existen riesgos importantes de la calidad del sistema que se acumulan. Usted puede externalizar el desarrollo y puede comprar el software de distribución masiva, pero no puede transferir los riesgos tan fácilmente.

Estos riesgos se derivan de cuatro áreas principales:

Acoplamiento: Fuerte interacción o consecuencia de la falla entre componente y sistema. Por ejemplo, considere un componente de base de datos en una aplicación Web. Si la base de datos falla, la aplicación falla.

Irreemplazabilidad: Pocos componentes similares disponibles. Un componente de base de datos es reemplazable, porque hay varios en el mercado. Pero el código, el cuál fue desarrollado a la medida no es normalmente reemplazable.

Esencialidad: Características clave en el sistema no son disponibles si la componente no funciona correctamente. Si usted tiene un applet12 en una aplicación que proporciona alguna característica deseable, adicional y falla, el sistema puede ser utilizado todavía. Pero por ejemplo una base de datos es usualmente esencial.

Problemas de calidad del fabricante: Alta probabilidad de la existencia de un componente erróneo. Si el vendedor no es confiable para construir un buen software o probarlo adecuadamente, entonces por supuesto que los riesgos aumentan.

¿Cómo podemos mitigar estos riesgos? Hemos utilizado cuatro métodos principales:

Uno de ellos es integrar, hacer el seguimiento y gestionar las pruebas del proveedor en un esfuerzo distribuido de las pruebas. Esto funciona para el desarrollo personalizado, usualmente no para Software Comercial de Distribución Masiva. Esto también implica una cierta relación que no podría existir entre el proveedor y el cliente.

Otro es confiar en las pruebas de componente del proveedor. Esto suena ingenuo, expresado de esta manera, pero lo hacemos todo el tiempo. Es una buena idea de por lo menos revisar sus referencias y su reputación de la industria antes de confiar en el proveedor.

Otro método consiste en corregir los problemas de las pruebas y los procesos de calidad del proveedor. Una vez más, usted tendrá que tener permiso para hacerlo.

Por último, usted puede hacer caso omiso a sus pruebas y reemplazarlas por completo, convirtiéndose en el laboratorio de facto de las pruebas del sistema.

Naturalmente, estos dos últimos escenarios son bastante costosos, consumen mucho tiempo, tienen muchas implicaciones negativas y políticas delicadas.

Por último, cuando usted esté probando sistemas integrados, recuerde de planificar usted mismo acerca de las pruebas de integración y sistema. Sólo porque un componente funciona de forma independiente no significa que funcione en el sistema.

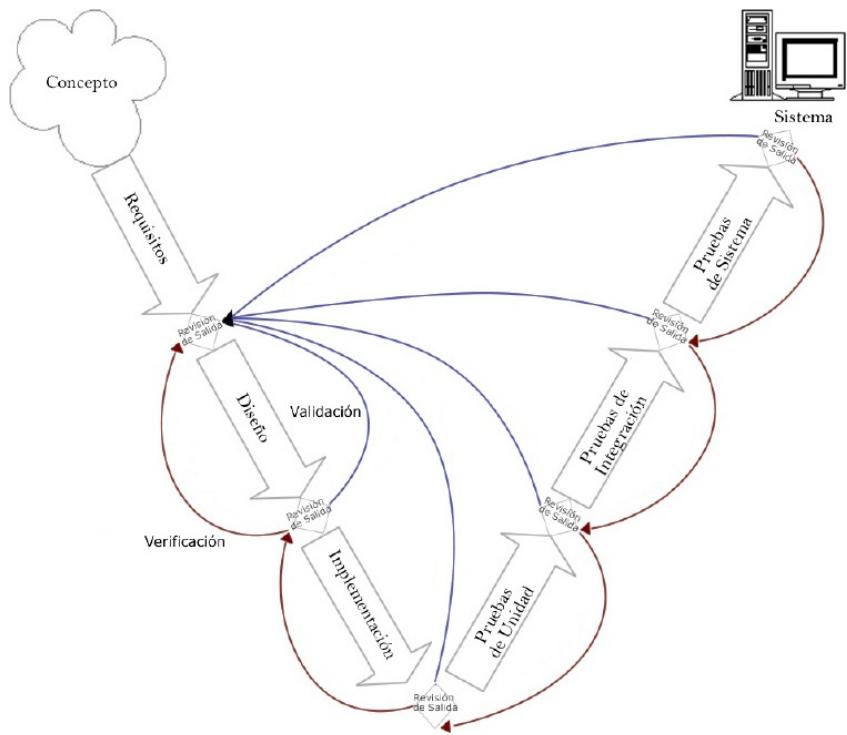


Figura 2.4: Verificación & Validación

Más allá de las revisiones y las actividades de las pruebas dinámicas como las pruebas unitarias y las pruebas de sistema, existen oportunidades para asegurar la calidad durante todo el ciclo de vida en cuanto a actividades adicionales de verificación y validación.

Glosario del ISTQB

Validación: Confirmación mediante un examen y a través de la provisión de evidencia objetiva de que los requisitos han sido cumplidos para un uso específico previsto o una aplicación.

Verificación: Confirmación mediante un examen y a través de la provisión de evidencia objetiva de que los requisitos especificados han sido cumplidos.

Hagamos una distinción entre estos dos tipos de actividades.

La verificación está buscando defectos en los entregables de las fases, comprobando que hemos seguido el proceso, y así sucesivamente. Pregúntese: "¿Estamos construyendo el sistema correctamente?" En otras palabras, la verificación es acerca de buenos procesos.

La validación es acerca de buscar defectos en el sistema, basada en los entregables de las fases, comprobando para ver si el sistema tiene calidad desde la perspectiva del cliente. Pregúntese: "¿Estamos construyendo el sistema correcto?" En otras palabras, la validación es acerca de buenos productos.

En la figura 2.4, puede ver que tenemos una revisión en la salida en el final de cada fase de un proyecto siguiendo un modelo V. En estas revisiones, **verificamos** que todas las acciones correctas han sido llevadas a cabo, basadas en los resultados finales de la fase anterior y nuestro modelo de proceso. **Validamos** que aún estemos en camino de entregar un sistema de calidad para el cliente.

A continuación hablemos del primero de los estándares referenciados en el programa de estudios básicos del ISTQB. Se trata del Estándar del Proceso de Software IEEE 12207.

Este estándar contiene las siguientes secciones:

1. Alcance, abordando el propósito, la aplicación, la adaptación, la conformidad y las limitaciones del estándar.
2. Referencias normativas.
3. Definiciones.
4. Aplicación del estándar a los procesos del ciclo de vida, incluyendo su adaptación y ajuste a la organización.
5. Los procesos principales del ciclo de vida para la adquisición, el suministro, el desarrollo, la operación y el mantenimiento.
6. Los procesos de apoyo tales como las publicaciones técnicas, la gestión de configuraciones, el aseguramiento de la calidad, la verificación y la validación independiente, las auditorías y la resolución de problemas.
7. Los procesos del ciclo de vida organizacional como la gestión, la infraestructura, el

mejoramiento y la capacitación.

El próximo estándar, es el modelo CMMI para la Madurez del Proceso de Software. El Modelo de Integración de Madurez de la Capacidad es un modelo de cinco niveles del Software Engineering Institute. Los cinco niveles son:

- Inicial: los proyectos son imprevisibles, controlados incorrectamente y reactivos.
- Gestionado: los procesos son establecidos en el nivel del proyecto, pero a menudo reactivamente.
- Definido: los procesos son establecidos a través de la organización y por lo general proactivamente.
- Gestionado cuantitativamente: los procesos son medidos y controlados en el nivel de la organización.
- Optimización: enfoque en la mejora continua, generalmente dirigido por datos.

Históricamente, las pruebas han sido destacadas muy poco en CMM y CMMI, conduciendo a modelos específicos de los procesos de pruebas como los Procesos de Pruebas Críticos y el Proceso de Evaluación, el Mejoramiento del Proceso de Pruebas ("TPI"), el Modelo de Madurez de Pruebas ("TMM") y T-Map.

El IEEE 12207 y CMMI, como la mayoría de los estándares, son los objetivos del aprendizaje del nivel K1, es decir, usted sólo tiene que recordar los estándares y su contenido. En el nivel básico, la meta del ISTQB es simplemente asegurar la familiaridad con la mayoría de los estándares.

Independientemente de los modelos del ciclo de vida y los modelos de madurez que usted seleccione, se aplican algunas características generales de las buenas pruebas.

Usted debería tener actividades de pruebas para cada actividad del desarrollo. Por ejemplo, las pruebas unitarias están vinculadas a la implementación. Los requisitos deberían ser revisados.

Los niveles de las pruebas tienen objetivos centrados, con coordinación para evitar los vacíos y la superposición. Esto se hace mejor en un documento de las políticas de las pruebas como se describió anteriormente.

El análisis y el diseño de pruebas se inician temprano para tratar de evitar los defectos.

Los probadores son involucrados en cualquiera de las revisiones en las que ellos tengan la calificación para atenderlas, así que ellos pueden brindar su perspectiva y mentalidad individual, como se describió en el capítulo anterior.

Usted puede libremente combinar o reorganizar los niveles de pruebas, para adaptar sus pruebas a los varios ciclos de vida y en resumen ser flexible acerca del método para las pruebas, con la condición de que usted mantenga estas características en la mente.

Glosario del ISTQB

Pruebas de robustez: Pruebas para determinar la robustez de un producto de software.

Robustez: El grado en el cual un componente o sistema puede funcionar correctamente en la presencia de entradas inválidas o condiciones estresantes del entorno. Véase también tolerancia de errores y tolerancia de faltas. Tenga en cuenta que este término no se citó específicamente para esta sección, pero se lo incluye aquí, porque es esencial para comprender el término pruebas de robustez.

2.1.1 Ejercicios

Ejercicio 1

El afamado experto en calidad W.E. Deming dijo, "todos los modelos son erróneos, algunos son útiles".

Indique cuál modelo del ciclo de vida en esta sección aplicó usted lo más cercano a su proyecto anterior (o, si no hubo modelo de organización, indique "codificar-y-corregir").

¿En qué medida piensa que el modelo fue útil?

¿En qué medida, si alguno, fue dañino?

Argumente.

Solución del Ejercicio 1

Un método cada vez más popular, seguido por más y más proyectos últimamente, es un modelo iterativo. El Proceso Unificado de Rational, es un método más analítico y apropiado para sistemas grandes de alto riesgo, mientras que las Metodologías Ágiles se enfocan en la rápida entrega de pequeños incrementos, lo cual es lo más seguro para proyectos de bajo riesgo. A causa de un gran número de componentes de software importantes disponibles—p. Ej., los sistemas de gestión de base de datos—, muchos proyectos implican la integración de sistemas, no solo el desarrollo de código nuevo.

¿Qué quiso decir Deming con su comentario? Un modelo de algo—un avión, una ciudad o un ciclo de desarrollo de software— está siempre erróneo porque no captura en su totalidad la cosa real que modela. Ningún modelo del desarrollo de software podría describir posiblemente cada cosa

que podría ocurrir en un proyecto real.

Un modelo de cualquier cosa es útil cuando le ayuda a entender qué hacer próximamente y cómo atacar al problema. Es útil cuando puede construir un framework, una perspectiva mental, para lo que está pasando y lo que debería pasar. Sin embargo, una vez cuando el modelo se convierte en un dogma rígido, lleva a equivocaciones y energía desperdiciada.

Ejercicio 2

Lea el Documento de Requisitos de Marketing de Omninet

¿Es más apropiado un modelo V o un modelo iterativo para el proyecto Omninet?

¿Son importantes el mantenimiento, la integración o la verificación y la validación para este proyecto?

Solución del Ejercicio 2

Mientras podría utilizar cualquiera, nosotros preferiríamos probablemente un modelo V. Los requisitos y el diseño del proyecto Omninet son sencillos, aunque la criticidad de una buena interfaz de usuario para los quioscos¹³ y el centro de llamadas demandaría muchos prototipos de interfaz de usuario y pruebas de usabilidad. Usted puede observar la utilización de los prototipos como iterativo. Un modelo V tradicional partidario podría indicar que los prototipos de interfaz de usuario son simplemente parte de los procesos de la verificación y la validación a través del ciclo de vida del software.

Ciertamente, nosotros propondríamos muchas pruebas de integración por ambos los desarrolladores y los probadores, porque todos los subsistemas importantes—el centro de datos, el centro de llamadas y el quiosco—cada uno consistirá en una serie de componentes. Algunos de esos componentes serán comprados probablemente en vez de ser construidos, así que las consideraciones de integración son importantes. Podría incluso decidir empezar a probar la integración entre los subsistemas antes de completar la integración de los subsistemas. Eso empieza a parecerse bastante a un modelo iterativo, a medida que le iría adicionando las características al centro de llamadas, el centro de datos y el quiosco, en incrementos, posteriormente probándolos juntos.

¿Ve usted cómo la distinción entre estos modelos se pone menos clara a menudo que examina un proyecto real y las decisiones que hay que tomar?

Una cosa de la que no tenemos que preocuparnos demasiado es el mantenimiento, como esto es una primera versión. Sin embargo, usted quiere probablemente probar la habilidad para instalar los parches, las actualizaciones, las definiciones de virus y software espía y etc. En otras palabras, no tenemos que probar una versión de mantenimiento durante la primera versión, pero tenemos que asegurarnos de que podemos publicar una versión de mantenimiento.

2.2 Niveles o Fases de Pruebas

Objetivos del Aprendizaje

LO-2.2.1 Comparar los diferentes niveles de pruebas: los objetivos principales, los objetos típicos de pruebas, los objetivos típicos de pruebas (p.ej. funcional o estructural) y los productos relacionados con el trabajo, la gente que prueba, los tipos de defectos y las fallas que deben ser identificadas. (K2)

Esta sección, Niveles o Fases de Pruebas, cubrirá los siguientes conceptos clave:

- Principales objetivos de las pruebas para cada nivel.
- Objetos típicos de prueba para cada nivel.
- Objetivos típicos de prueba para cada nivel.
- Productos del trabajo de pruebas para cada nivel.
- Participantes de las pruebas para cada nivel.
- Tipos de defectos y fallas para cada nivel.

Empecemos con el nivel de pruebas de componente, a veces llamado pruebas de unidad.

Anteriormente, habíamos mencionado que los objetivos tienden a variar según los diferentes niveles de pruebas. Para las pruebas de componente, los objetivos típicos son: encontrar los defectos, construir la confianza y reducir el riesgo en las partes individuales del sistema bajo prueba antes de la integración de sistemas.

Las pruebas de componente se basan generalmente en el código y la base de datos. A menudo existe una referencia a las especificaciones de los requisitos y el diseño, al menos como oráculos de pruebas. (Por cierto, un oráculo de prueba, de manera informal, es lo que se consulta para averiguar el resultado correcto o esperado de una prueba). Al realizar las pruebas basadas en los riesgos—acerca de lo que hablaremos en detalle más adelante—los riesgos de calidad también serán considerados.

Las pruebas de componente pueden incluir los tipos de pruebas así como los de comportamiento o caja negra (p.ej. los de funcionalidad, utilización de recursos y rendimiento) y los tipos de pruebas estructurales o de caja blanca.

Glosario del ISTQB

Pruebas de rendimiento: El proceso de pruebas para determinar el rendimiento de un producto de software. Véase también las pruebas de eficiencia.

Rendimiento: El grado en el cual un sistema o componente cumple sus funciones designadas dentro de las restricciones dadas en relación con el tiempo del procesamiento y la tasa de transferencia. Véase también la eficiencia. Tenga en cuenta que este término no se citó específicamente para esta sección, pero se lo incluye aquí, porque es esencial para comprender el término pruebas de rendimiento.

Pruebas estructurales: Véase pruebas de caja blanca.

Pruebas basadas en los riesgos: Un método para las pruebas para reducir el nivel de los riesgos del producto e informar a los interesados del negocio acerca de su estado, comenzando en las etapas iniciales de un proyecto. Esto involucra la identificación de los riesgos del producto y la utilización de los niveles de riesgos para guiar el proceso de pruebas.

Pruebas de caja blanca: Las pruebas basadas en un análisis de la estructura interna del componente o sistema.

El objeto de prueba o ítem sometido a prueba varía. Los puristas en pruebas dirían que las pruebas unitarias deberían abordar el ítem independiente más pequeño y comprobable. Para los lenguajes de procedimientos como C o COBOL es la función o el procedimiento. Para los lenguajes orientados a objetos como Java o C++ es la clase. Sin embargo, en la práctica real, la gente suele realizar las llamadas “pruebas unitarias” de partes más grandes del sistema.

La frase “prueba de componente” es aún más confusa en su utilización en el mundo real. Esto puede significar pruebas de un componente distinto que proporciona algunos servicios a otras componentes. Esa componente puede estar formada por unidades, las cuales han sido probadas con las pruebas de unidad.

Entonces, en la práctica, las frases “pruebas de componente” y “pruebas de unidad” pueden aplicarse a las pruebas de una variedad de objetos de prueba, que incluyen componentes, programas, utilitarios de conversión de datos y de migración, módulos de bases de datos e incluso campos individuales de entrada, pantallas individuales de entrada o grupos de pantallas.

Ya sea que las llamemos pruebas de unidad o de componente, hay una necesidad de los arneses y las herramientas, porque el sistema no es en sí mismo completo ni independientemente comprobable. Entonces, tendemos a encontrar la utilización de arneses en el nivel de la interfaz de programación de aplicaciones denominados drivers y stubs. Hay herramientas disponibles para ambos tanto gratuitas como comerciales.

Glosario del ISTQB

Driver: Un componente de software o herramienta de pruebas que reemplaza un componente que se encarga del control y/o la invocación de un componente o sistema.

Stub: Una implementación esquelética o de propósito especial de un componente de software, que puede ser utilizada para desarrollar o probar otro componente que invoca o que es de otra manera dependiente del stub. Éste sustituye a un componente invocado.

Ahora, en cuanto a quién es responsable, por lo general son los programadores, pero el nivel de capacidad y el grado de ejecución varía. Incluso los partidarios de las metodologías de pruebas ágiles, que elogian las virtudes de las buenas pruebas unitarias automatizadas, a menudo muestran una carencia de conciencia y rigor más bien alarmante con respecto a las mejores prácticas de las pruebas.

Hablando de metodologías, el proceso de pruebas de componente o unitarias podría ser analizado en este punto.

Normalmente hay algunas cosas que son por lo general verdad y típicas acerca de las pruebas unitarias o de componente. Por un lado, implica el acceso al código. Por lo general, son ejecutadas en un entorno de desarrollo, por el programador que escribió el código. Y, como se explicó anteriormente, se requieren drivers, stubs o arneses, porque la unidad no es independiente.

Se da el caso también, por lo general, de que el programador corrige los defectos encontrados sobre la marcha. En otras palabras, los defectos son corregidos cuando son encontrados sin ningún tipo de informe. Esto parece ser más eficiente en un principio, pero reduce la transparencia del proceso del desarrollo con respecto a la calidad. La mayoría de las organizaciones tienen poco o ningún conocimiento acerca de los niveles reales de la calidad durante la implementación y las pruebas unitarias, porque no miden los defectos encontrados y corregidos en las revisiones del código y las pruebas de unidad. Esto significa que las organizaciones no pueden tomar decisiones racionales, dirigidas por datos acerca de la calidad y el grado apropiado de las pruebas, lo que explica por qué muchos jefes de proyectos se sorprenden—y no tan gratamente— del nivel de calidad cuando se inician las pruebas más formales.

Hay una forma de pruebas unitarias que se llama desarrollar primero las pruebas (“Test First Development-TFD”) o desarrollo dirigido por las pruebas (“Test Driven Development-TDD”). Parece que esto vuela al revés la forma de codificación, en la cual el programador desarrolla primero un conjunto de pruebas unitarias, luego construye e integra el código necesario para que esas

pruebas pasen. A continuación él procede a adicionar más pruebas unitarias correspondientes a funciones adicionales y repite el proceso.

Glosario del ISTQB

Desarrollo dirigido por pruebas: Una forma de desarrollo de software donde los casos de prueba son desarrollados y a menudo automatizados, antes de que el software sea desarrollado para ejecutar aquellos casos de prueba.

Si bien esto podría parecer revolucionario, es en realidad sólo un giro en las mejores prácticas de la programación que se remontan a décadas pasadas. Cuando trabajábamos como programadores al principio de los años 80, la regla siempre fue, codificar un poco, probar un poco, codificar un poco más, probar un poco más, etc. Este método fue posible por medio de la liberación del programador de las tarjetas perforadas, los compiladores y los enlazadores más rápidos que se ejecutaban en terminales. Todo lo viejo es nuevo otra vez, como dice el refrán.

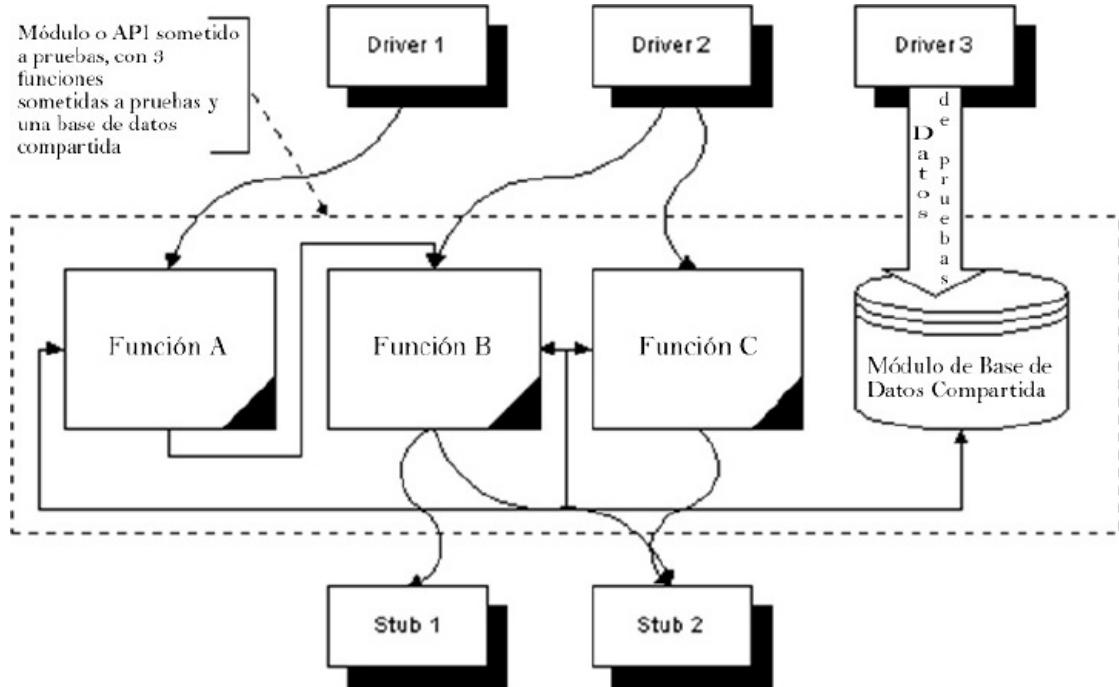


Figura 2.5: Drivers y Stubs

Hemos mencionado los drivers, los stubs y los arneses unas cuantas veces, así que tal vez algunas definiciones están claras.

Como hemos mencionado antes, durante las pruebas unitarias, de componente y de integración—y, de hecho, en cualquier momento que estemos probando directamente una interfaz de programación de aplicaciones (API) — a menudo es necesario simular las partes de la estructura general del programa por encima o por debajo del módulo o los módulos bajo prueba. Por cierto, por el módulo nos referimos ya sea a una función (si estamos realizando las pruebas de programas que contienen procedimientos) o una clase (si estamos realizando las pruebas de programas que son orientados a objetos).

También podría ser necesaria la configuración de datos.

Para esto, tenemos dos tipos de arneses. Uno se llama driver, el cual consiste en una función o funciones (en una clase o clases), que llaman o invocan o utilizan el módulo o los módulos bajo pruebas. El otro se llama stub, el cual consiste en una función o funciones (en una clase o clases) que son invocados —directa o indirectamente— por el módulo o los módulos bajo pruebas.

En el mundo orientado a objetos, usted escuchará el término “objeto de simulación” con frecuencia para referirse a los arneses de pruebas.

Continuemos con el nivel de pruebas de integración.

Una vez más, los objetivos pueden variar, pero normalmente queremos encontrar defectos, construir la confianza y reducir el riesgo en las relaciones e interfaces entre los pares y grupos de componentes en el sistema bajo prueba a medida que las partes se unen.

Puesto que estamos examinando las relaciones e interfaces, las pruebas de integración se basan típicamente en el diseño del sistema, la arquitectura, los flujos de trabajo, los casos de uso, los esquemas de la base de datos y los flujos de datos. Otra vez, si estamos realizando las pruebas basadas en los riesgos, utilizaremos adicionalmente los riesgos de calidad para guiar nuestras pruebas.

Las pruebas de integración pueden incluir varios tipos de pruebas así como las pruebas de comportamiento (p.ej. las de funcionalidad, utilización de recursos y rendimiento). Éstas pueden

incluir las pruebas estructurales como los flujos de llamadas y los flujos de datos.

El objeto de prueba o ítem sometido a prueba es una colección de unidades (o componentes) a menudo referida como una “construcción” (“build”) o en algunos casos, una “red central” (“backbone”). Estas construcciones pueden incluir una implementación de la base de datos de un subsistema, una infraestructura, interfaces, una configuración del sistema y datos de configuración. La selección de las unidades o los componentes, para que sean incluidos una construcción, es influenciada por el hecho de que las pruebas de integración consisten en probar las interfaces entre los componentes, las interacciones con diferentes partes de un sistema (como el sistema operativo, el sistema de archivos y el hardware), y las interfaces entre las unidades, componentes o sistemas.

Al igual que las pruebas unitarias y de componente, las pruebas de integración requieren con frecuencia arneses y herramientas, debido a que las construcciones no son siempre independientes y comprobables. Estos arneses pueden actuar en el nivel de la interfaz de programación de aplicaciones (API). También pueden estar en el nivel de la interfaz de línea de comandos.

A veces, si estamos probando en la interfaz de usuario, las herramientas de interfaz gráfica de usuario son útiles, pero eso es algo raro. Hay herramientas tanto gratuitas como comerciales necesarias para las pruebas de integración.

En cuanto a quién es el responsable, tanto los probadores como los programadores colaboran idealmente. Desafortunadamente, a menudo nadie es responsable. Los probadores no siempre tienen las habilidades técnicas necesarias para diseñar y ejecutar las pruebas de integración, y a los programadores no siempre se les da el tiempo y la indicación para ayudar. Esto da lugar a muchas dificultades durante las primeras semanas de las pruebas del sistema, debido a que la primera parte del período de ejecución de la prueba del sistema se convierte de hecho en una prueba de integración big-bang¹⁴. Más adelante veremos más acerca de la prueba de integración big-bang.

- **Big bang**
 - Toma todos los módulos probados; los pone a todos juntos; los prueba
 - Rápido, ¿Pero, dónde está el defecto?
 - ¿Por qué esperar hasta que todo el código este escrito para iniciar la integración?
- **Ascendente**
 - Comienza con los módulos de la capa inferior; utiliza drivers adecuados; los prueba
 - Repite el proceso, reemplazando los drivers con módulos, hasta que termine
 - Buen aislamiento de defectos, ¿Pero qué pasa si los problemas desagradables están arriba?
- **Descendente**
 - Al igual que la ascendente, pero comienza de la parte superior y utiliza stubs
 - Buen aislamiento de defectos, ¿Pero qué pasa si los problemas desagradables están abajo?
- **Funcional o transaccional**
 - Integra el conjunto de componentes necesarios para implementar una función o transacción individual
 - Repite el proceso, reemplazando los stubs y drivers con el conjunto de componentes para implementar la siguiente función o transacción
 - Buen aislamiento de defectos, puede encontrar defectos de integración con respecto al orden de los riesgos

Figura 2.6: Técnicas de Integración

¿De dónde vienen estas construcciones y cómo son ensambladas? Existen varias técnicas.

La primera técnica no es realmente una técnica en absoluto, es tanto como alguna especie de ejercicio de ingeniería de software en ilusiones. Es denominada irónicamente “integración big bang”. Consiste en tomar un conjunto de (probadas ojalá) unidades, componentes, clases, funciones, cualquier cosa y ponerlos todos juntos al mismo tiempo, luego probarlos. Esto parece rápido, pero ¿Qué sucede cuando usted encuentra un defecto? Mejor aún, ¿Qué pasa cuando ni siquiera se puede producir un sistema ejecutable de la colección?

Básicamente, éste es un ejemplo de la peor práctica de pruebas en acción. Es una peor práctica de pruebas para planificar, diseñar o ejecutar las pruebas acerca del supuesto de que todas las pruebas pasarán y todo estará bien. Esa no es la forma en que generalmente funciona.

Además, ¿Por qué esperar hasta que todo el código este escrito para iniciar la integración? ¿No hay un principio en las pruebas que diga que las pruebas tempranas son buenas?

Entonces, supongamos que empezamos a construir las capas más bajas del sistema, normalmente las que se comunican con el sistema operativo, el hardware y la base de datos. Esto, por supuesto, involucra drivers, porque tenemos que llamar a estas capas. Construimos cada capa, de forma

incremental, y la probamos. A medida que construimos las capas, los drivers son reemplazados por los módulos reales y son escritos nuevos drivers para aquellos nuevos módulos introducidos. Repetimos este proceso hasta que hayamos terminado la integración.

Esta técnica proporciona un buen aislamiento de defectos, a diferencia de la técnica de big bang. Sin embargo, ¿Qué pasa si los problemas desagradables están en la capa superior? No los encontraremos hasta el final.

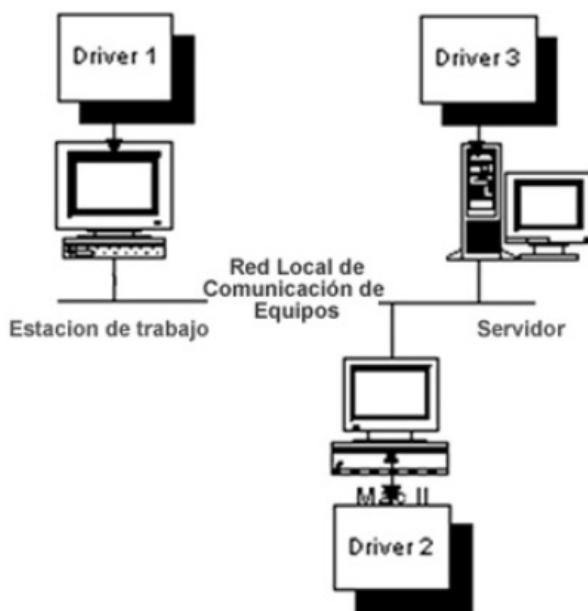
Eso viola una de nuestras reglas heurísticas para las pruebas: Encontrar primero las cosas que asustan. Deberíamos intentar de encontrar los problemas en orden de prioridad, donde sea posible. Así, si sospechamos que problemas serios podrían acechar en la capa superior, generalmente en la interfaz de usuario, mejor no esperemos hasta el final.

Esto significa que podríamos querer empezar a construir con las capas más altas del sistema, lo cual, otra vez, es usualmente la interfaz de usuario. Por supuesto, esto implica stubs, porque las capas inferiores no existen. Como antes, construimos incrementalmente cada capa y las probamos. A medida que creamos las capas, los stubs son reemplazados por los módulos reales y los nuevos stubs son escritos para esos nuevos módulos introducidos. Repetimos este proceso hasta que hayamos terminado la integración.

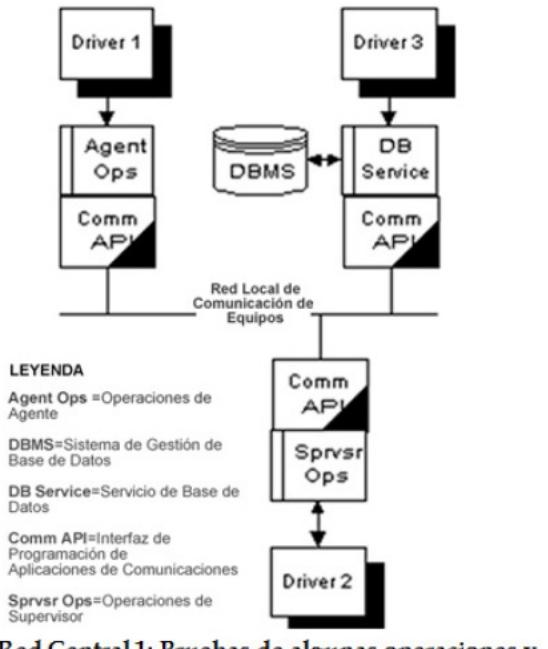
Otra vez, tenemos buen aislamiento de defectos. Sin embargo, si encontramos defectos graves en la capa inferior, estamos en graves problemas.

Vea, el problema aquí es que estamos permitiendo que la arquitectura del sistema determine el orden de las pruebas. Ahora, por supuesto, durante las pruebas de integración, la arquitectura es muy importante, pero no podemos contar con la arquitectura para que tome nuestra mano y nos muestre los defectos.

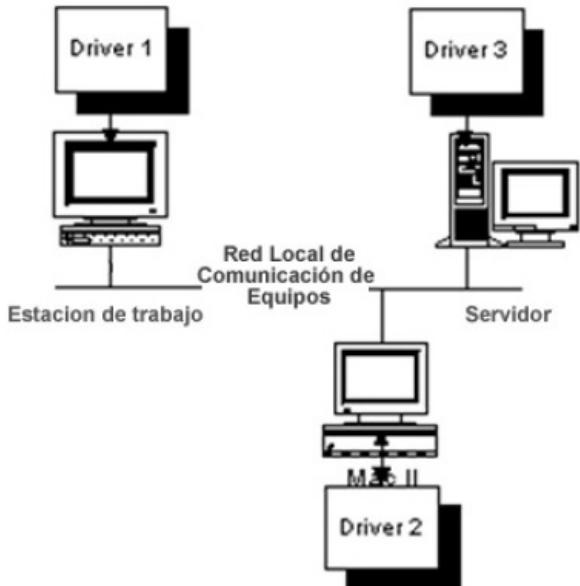
En lugar de utilizar la arquitectura, a veces la gente trata de utilizar la funcionalidad. La integración funcional implica la identificación de los módulos necesarios para apoyar una determinada característica, luego construirlos y probarlos. Identificar otra característica, añadir los módulos necesarios, probar, y así sucesivamente, hasta que esté terminado. Esto nos libera en alguna forma extensa las equivocaciones potenciales inherentes en las técnicas descendentes y ascendentes, pero ¿Qué pasa si tomamos las funciones equivocadas para empezar?



Red Central 0: Pruebas del software de comunicación básico y de la arquitectura de redes



Red Central 1: Pruebas de algunas operaciones y servicios principales a través de la API y las redes de comunicación



Red Central N: Las pruebas del sistema entero completamente integrado de extremo a extremo. Cuando esto funciona, entonces estamos listos para la Prueba de Sistema

Figura 2.7: Integración de Red Central (“Backbone Integration”)

La integración de red central (“Backbone Integration”) es un medio de hacer frente a esto. En la integración de red central, utilizamos el riesgo y la arquitectura y las características para que nos guíen.

Identificamos un conjunto de módulos críticos que trabajan juntos, que pensamos que es probable que tengan un número significativo de defectos o defectos significativos en general y que mantienen una o varias características clave. Construimos una red inicial, que contiene aquellos módulos, y luego utilizamos drivers y stubs para probarlos. Una vez que tenemos esa red trabajando, repetimos el proceso en el orden de los riesgos, otra vez reemplazando stubs y drivers con módulos a medida que avanzamos. Este método proporciona un aislamiento bueno de defectos, un orden lógico de las características que deben ser probadas, y tiene la mayor probabilidad de encontrar defectos de integración en el orden de importancia.

Las pruebas de integración pueden ocurrir en múltiples niveles, en otras palabras, no es inusual para un cierto proyecto complejo de requerir más de un nivel de pruebas de integración. Podría haber pruebas de integración de componentes seguidas o en paralelo de un nivel de pruebas de unidad o componente y luego pruebas de integración de sistema, donde estemos probando las interacciones entre sistemas completos seguidos o en paralelo de un nivel de pruebas de sistema.

Las pruebas de integración de sistema son particularmente complejas. Los sistemas podrían haber sido desarrollados por diferentes organizaciones e incluso en diferentes períodos de tiempo, con

tecnologías e interfaces diferentes y tal vez no completamente compatibles. Cuando hay múltiples organizaciones involucradas, diferentes organizaciones pueden controlar las interfaces de los sistemas, lo cual hace que los cambios en los sistemas sean al mismo tiempo más peligrosos y, paradójicamente, menos probables de ser correctamente gestionados.

Debido a que los procesos de negocio podrían atravesar sistemas, los problemas que surgen cuando los cambios dañan una interfaz —y así un proceso de negocio—puede tener impactos críticos en la organización.

Finalmente, además de los problemas de compatibilidad del sistema, pueden surgir problemas de compatibilidad en el hardware. Los sistemas anfitriones, cuando están conectados juntos, podrían no funcionar correctamente debido a cuestiones en el nivel del hardware. Tales problemas podrían estar más allá de la capacidad de la organización para resolverlos.

Continuemos con el nivel de pruebas de sistema.

Otra vez, los objetivos pueden variar, pero típicamente queremos encontrar los defectos, construir la confianza y reducir el riesgo en los comportamientos generales y particulares, las funciones y las respuestas del sistema sometido a pruebas en su totalidad. Estamos mirando todo el sistema, por lo que las pruebas de sistema se basan generalmente en los requisitos del sistema, el diseño de alto nivel, los casos de uso y la experiencia del usuario y probador con sistemas similares. A veces las organizaciones tienen listas de comprobación de las características principales de calidad que deben ser abordadas durante las pruebas de sistema y también pueden ser necesarios la cobertura de los centros de datos o los entornos del cliente. Otra vez, si estamos realizando pruebas basadas en los riesgos, utilizaremos riesgos de calidad para guiar nuestras pruebas también.

Las pruebas de sistema pueden incluir los tipos de pruebas de comportamiento, así como de funcionalidad, seguridad, fiabilidad, usabilidad, portabilidad y rendimiento. Las técnicas de pruebas estructurales como la cobertura de sentencia, rama y bucle son utilizadas a veces para comprobar la completitud de las pruebas de comportamiento.

Glosario del ISTQB

Pruebas de Portabilidad: El proceso de pruebas para determinar la portabilidad de un producto de software.

Portabilidad: La facilidad con la cual el producto de software puede ser transferido de un entorno de hardware o software a otro. Tenga en cuenta que este término no ha sido enunciado específicamente en esta sección, pero se lo incluye aquí, porque es esencial para comprender el término pruebas de portabilidad.

Pruebas de fiabilidad: El proceso de pruebas para determinar la fiabilidad de un producto de software.

Fiabilidad: La capacidad del producto de software para realizar sus funciones necesarias en las condiciones establecidas por un período de tiempo especificado, o para un número de operaciones especificadas. Tenga en cuenta que este término no ha sido enunciado específicamente en esta sección, pero se lo incluye aquí, ya que es esencial para comprender el término pruebas de fiabilidad.

Pruebas de seguridad: Pruebas para determinar la seguridad del producto de software. Véase también pruebas de funcionalidad.

Seguridad: Atributos de los productos de software que influyen en su capacidad para prevenir el acceso no autorizado, ya sea accidental o intencionado, a los programas y datos. Véase también la funcionalidad. Tenga en cuenta que este término no ha sido enunciado específicamente en esta sección, pero se lo incluye aquí, porque es esencial para comprender el término pruebas de seguridad.

Pruebas de usabilidad: Las pruebas para determinar la medida en la que el producto de software es comprendido, fácil de aprender, fácil de operar y atractivo para los usuarios en determinadas condiciones.

El objeto de prueba o ítem sometido a pruebas es idealmente todo el sistema, en un entorno de pruebas, el cual es una réplica de la producción tan realista como sea posible. Esto debería incluir cualquier documentación, manuales o ayuda en línea entregada con el sistema. También debería incluir datos de configuración.

Glosario del ISTQB

Entorno de pruebas: Un entorno de pruebas que contiene el hardware, la instrumentación, los simuladores, las herramientas de software y otros elementos de soporte necesarios para realizar una prueba.

Para las pruebas de sistema, una gran variedad de herramientas están disponibles, para trabajar en la interfaz de programación de aplicaciones (API), la interfaz de línea de comandos y la interfaz gráfica de usuario. A diferencia de los anteriores niveles de pruebas, el sistema es independiente y comprobable. El software gratis o software comercial existen como herramientas necesarias para las pruebas de sistema.

En cuanto a quién es responsable, las pruebas de sistema son típicamente el reino del probador independiente. Los usuarios también podrían estar involucrados.

Hablando de los usuarios como probadores, pasamos al nivel de pruebas de aceptación.

Aquí, el objetivo está típicamente enfocado en construir la confianza, para demostrar que el producto o sistema está listo para el despliegue o la versión. Los niveles de pruebas de aceptación generalmente no tienen como objetivo la búsqueda de defectos.

Ésta parece una afirmación obvia y sencilla, pero aquí surgen falsas expectativas. Una empresa para la cual RBCS/Business Innovations hizo una evaluación tenía las pruebas de aceptación ejecutándose en paralelo con las pruebas de sistema, en el mismo software y en el mismo entorno

de pruebas. Los usuarios que realizaban las pruebas de aceptación se molestaron cuando encontraron defectos. Nosotros preguntamos, ¿Cómo podrían esperar ellos razonablemente no encontrar defectos en el mismo software que estaba siendo probado activamente para encontrar defectos al mismo tiempo?

Entonces, aquí hay una lección clave para las pruebas de aceptación: Si el objetivo es **no** encontrar ningún defecto, entonces las pruebas de aceptación se deben ejecutar después de que todos los niveles de pruebas previamente planificados hayan sido completados, incluyendo la corrección de los defectos para esos niveles.

En otra ocasión, recibimos una llamada telefónica de un cliente de RBCS/Business Innovations preguntando acerca de la cobertura de las pruebas de aceptación. Sus usuarios quisieron probar las condiciones que previamente no habían sido probadas en las pruebas de sistema y no revelarían a la organización TI lo que pretendían hacer durante las pruebas de aceptación. Al mismo tiempo, los usuarios estaban diciendo que sería un verdadero problema si se encontraban defectos. Preguntamos ¿Cómo podrían esperar los usuarios no encontrar ningún defecto si ellos no informarían a la organización TI qué probar antes de entregar el software?

Entonces, aquí esta otra lección clave para las pruebas de aceptación: Si el objetivo es **no** encontrar ningún defecto, entonces, las pruebas de aceptación deberían cubrir un subconjunto de las condiciones probadas en los niveles anteriores de pruebas.

Al igual que con las pruebas de sistema, estamos examinando todo el sistema, por lo que las pruebas de aceptación se basan típicamente en los requisitos del sistema. Esto puede incluir los requisitos de los usuarios, los requisitos del sistema, los casos de uso, los procesos de negocio y los informes del análisis de los riesgos. Las pruebas de aceptación también podrían tener en cuenta las obligaciones contractuales si el software está siendo comprado, construido a medida, o adquirido. La experiencia de los usuarios y los probadores con sistemas similares está involucrada algunas veces, aunque ésta puede ser una espada de doble filo. Hemos visto situaciones en las que dar rienda suelta a los usuarios durante las pruebas de aceptación lleva a la re-definición retroactiva de los requisitos del proyecto y eventualmente al fracaso del proyecto.

Incluso si estamos llevando a cabo pruebas basadas en los riesgos, los riesgos de calidad generalmente juegan un rol más limitado en la guía de nuestras pruebas durante las pruebas de aceptación. Las consideraciones del impacto en los negocios podrían determinar qué pruebas ejecutamos, pero la secuenciación de las pruebas y la reevaluación de los niveles de los riesgos de calidad basados en los resultados de las pruebas son menos probables, porque los resultados de las pruebas (al menos en teoría) sólo van a confirmar que el sistema funciona.

Las pruebas de aceptación están típicamente abordando los tipos de pruebas de comportamiento, especialmente de funcionalidad. Otras pruebas como de compatibilidad, rendimiento y seguridad también pueden ser importantes. Los tipos de pruebas estructurales no se utilizan normalmente, porque los usuarios ven el sistema como una caja negra.

Al igual que con las pruebas de sistema, el objeto de pruebas o ítem sometido a prueba es todo el sistema. A veces es probado en producción o en el entorno del cliente, sin embargo una práctica más segura es utilizar un entorno de pruebas. Los probadores deberían tener especificaciones —o por lo menos un conocimiento sólido de—los procesos de negocio, operacionales y de mantenimiento, los procedimientos de usuarios, los formularios y los informes. El entorno de pruebas debería incluir datos de configuración apropiados y realistas así como también una réplica —si no tan exacto una copia—de los datos de producción.

Para las pruebas de aceptación, cuando se utilizan herramientas, estas por lo general son aplicadas en la interfaz gráfica del usuario. Las excepciones podrían incluir el uso de los generadores de carga para el rendimiento o pruebas de simulación de carga.

Glosario del ISTQB

Pruebas de carga: Un tipo de pruebas de rendimiento conducidas para evaluar el comportamiento de un componente o sistema con una carga cada vez mayor, p.ej. número de usuarios concurrentes y/o número de transacciones, para determinar qué carga puede ser controlada por el componente o sistema. Véase también pruebas de rendimiento y pruebas de estrés.

La pruebas de aceptación a menudo son realizadas por los usuarios y los clientes, aunque algunos de nuestros clientes tienen probadores independientes para guiar y asistir a los usuarios. Ésta es una buena manera de mantener a los usuarios enfocados en las condiciones de pruebas legítimas, también, ayudando a evitar que divaguen en áreas no especificadas y re-definan el significado de la palabra “terminado” para el proyecto.

Las pruebas de aceptación presentan una amplia variación en el mundo real.

Vemos usualmente que las organizaciones realizan pruebas de aceptación de usuario, donde los usuarios de negocios verifican la aptitud para los propósitos funcionales y para las aplicaciones clave de negocios.

Menos común pero no desconocido es el nivel de pruebas operacionales, donde el objetivo es la aceptación por los administradores de sistemas. Como ejemplos tenemos las copias de seguridad-restauración, la recuperación de desastres, la gestión de usuarios, el mantenimiento, la carga y la

migración de datos y la seguridad.

Por cierto, si usted está en una industria relacionada con la defensa, el uso del término “pruebas operacionales” puede ser confuso. En los proyectos de defensa, “las pruebas de desarrollo” se refieren a las pruebas durante el desarrollo del sistema, a menudo por el contratista de la defensa. Las “pruebas operacionales” se refieren a lo que es, básicamente, las pruebas de sistema y las pruebas de integración de sistemas, a menudo por los usuarios reales.

El Departamento de Defensa de los Estados Unidos mantiene una gran instalación en Arizona en Fort Huachuca denominado el Comando Conjunto de Pruebas de Interoperabilidad que se especializa en las pruebas de integración de sistemas.

Glosario del ISTQB

Pruebas de interoperabilidad: El proceso de pruebas para determinar la interoperabilidad de un producto de software. Véase también pruebas de funcionalidad.

Interoperabilidad: La capacidad del producto de software para interactuar con uno o más sistemas o componentes especificados. Tenga en cuenta que este término no ha sido enunciado específicamente en esta sección, pero se lo incluye aquí, ya que es esencial para la comprensión del término pruebas de interoperabilidad.

Cuando el software es comprado o construido con respecto a las especificaciones, usualmente observamos las pruebas de contrato y cuando están involucradas las regulaciones de gobierno o de la industria, observamos las pruebas de reglamentación. Cada una implica la verificación de la conformidad con los requisitos acordados contractualmente o legalmente encargados, los reglamentos o los estándares.

En el mundo del software para el mercado público, también observamos las pruebas alfa y beta, mientras que las organizaciones TI que desarrollan software para el uso interno tienen un equivalente denominado pruebas de campo o pruebas piloto. Éstas son formas de pruebas de aceptación en las que efectivamente podríamos desear encontrar defectos. Sin embargo los objetivos son mayormente construir la confianza de los clientes o los usuarios potenciales o existentes. Las pruebas beta, las pruebas piloto y las pruebas de campo se realizan en el entorno real o los entornos en los cuales el sistema será desplegado.

Glosario del ISTQB

Pruebas alfa: Pruebas operacionales simuladas o reales por los usuarios/clientes potenciales o por un equipo de pruebas independiente en el sitio de los desarrolladores, pero fuera de la organización de desarrollo. Las pruebas alfa son a menudo empleadas para el software comercial de distribución masiva como una forma de pruebas de aceptación internas.

Pruebas beta: Pruebas operacionales por los usuarios/clientes potenciales y/o existentes en un sitio externo sin involucrar de alguna forma a los desarrolladores, para determinar si un componente o sistema satisface las necesidades del usuario/cliente y se ajusta a los procesos del negocio. Las pruebas beta se emplean a menudo como una forma de pruebas de aceptación externa para el software comercial de distribución masiva con el objeto de obtener retroalimentación del mercado.

Pruebas de campo: Véase pruebas beta.

Las pruebas piloto implican típicamente una muestra de la comunidad de los usuarios y podrían incluir pruebas paralelas, donde el trabajo se realiza con el nuevo sistema y al mismo tiempo, el mismo trabajo se realiza en el sistema existente, comparando resultados. Esto ayuda a reducir el riesgo de despliegue. Obviamente, la comunidad de los usuarios debe ser seleccionada para ser representativa, si se desea gestionar eficazmente los riesgos.

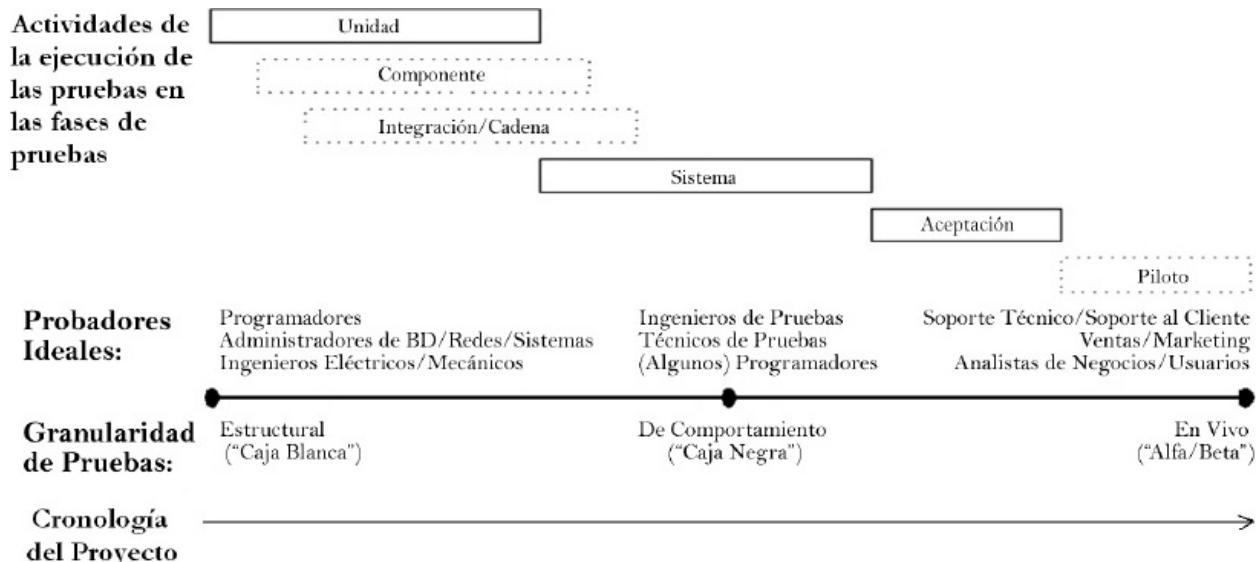


Figura 2.8: Pruebas Dominantes: Tempranas, Multidisciplinarias

Nos hemos referido antes en este libro al concepto de las pruebas como un filtro. Mediante el uso de una secuencia de filtros, todos bien alineados en cuanto a la cobertura y la secuencia, podemos

conseguir un entregable de muy alta calidad.

Esto es parte de lo que nos referimos como pruebas dominantes¹⁵(“pervasive testing”), acerca de las cuales explicaremos más en un momento.

En esta figura, se ve la secuencia del tiempo de los niveles progresivos de las pruebas. Los niveles mostrados en líneas punteadas podrían o no podrían estar presentes, dependiendo del proyecto. Aquí suponemos que estamos trabajando en un proyecto de ciclo de vida secuencial, ya que las pruebas del sistema no se inician hasta que las pruebas unitarias son completas. En un ciclo de vida iterativo, las pruebas de sistema se inician tan pronto como las primeras unidades de incremento se completan y se prueban con pruebas de unidad.

También se puede observar la participación de los cambios y la responsabilidad para los niveles.

Debajo de eso, se muestran la granularidad típica y los tipos generales de pruebas. Los tipos estructurales de las pruebas son los más importantes durante los niveles más tempranos de las pruebas, mientras que las pruebas del tipo de comportamiento predominan en los niveles medio y más altos. Las pruebas de aceptación son pruebas de comportamiento en vivo.

¿Cuál es el valor de las pruebas dominantes? Otra vez, la idea principal es conseguir el entregable “más puro” posible, con el porcentaje máximo de defectos eliminados, de la manera más económica y rápida.

Para hacer esto, utilizamos diferentes participantes en las diferentes granularidades de las pruebas. Esto se debe a que se necesitan diferentes habilidades en estas diferentes granularidades de pruebas.

Tenemos diferentes granularidades predominantes en cada fase o nivel de pruebas. Las pruebas unitarias, que se centran en pequeñas partes del sistema, son principalmente estructurales. Las pruebas de sistema, que se ocupan del sistema general, son principalmente de comportamiento. Las pruebas de aceptación, que se ocupan con el uso en el mundo real, son principalmente datos en vivo y flujos de trabajo del mundo real.

La manera de pensar acerca de esto es como una secuencia de filtros con porosidad diferente, cada una capaz de eliminar efectivamente los defectos más probables de ser encontrados en ese nivel de pruebas.

Sin embargo un problema que hemos visto en nuestro trabajo con clientes es el archivo del trabajo de las pruebas en silos cuando se usan los diferentes niveles de pruebas. Vacíos y superposiciones evolucionan y el pensamiento se hace dogmático y orientado a la culpa.

Así, al realizar las pruebas dominantes, no olvide de permanecer flexible. Las técnicas de pruebas de varias granularidades pueden ser útiles en todas las fases de la ejecución de pruebas, aunque la mezcla cambiará. La cantidad de solapamiento de nivel o fase es dirigida por los criterios de entrada y salida, los cuales deberían ser definidos de una manera que sea apropiada para el proyecto, la organización, el ciclo de vida, y así sucesivamente. Y, usted no debe sentirse obligado a utilizar un nivel de pruebas o fase en particular sólo porque el programa de estudios básico del ISTQB lo menciona. No todos los niveles de pruebas ocurren en todos los proyectos.

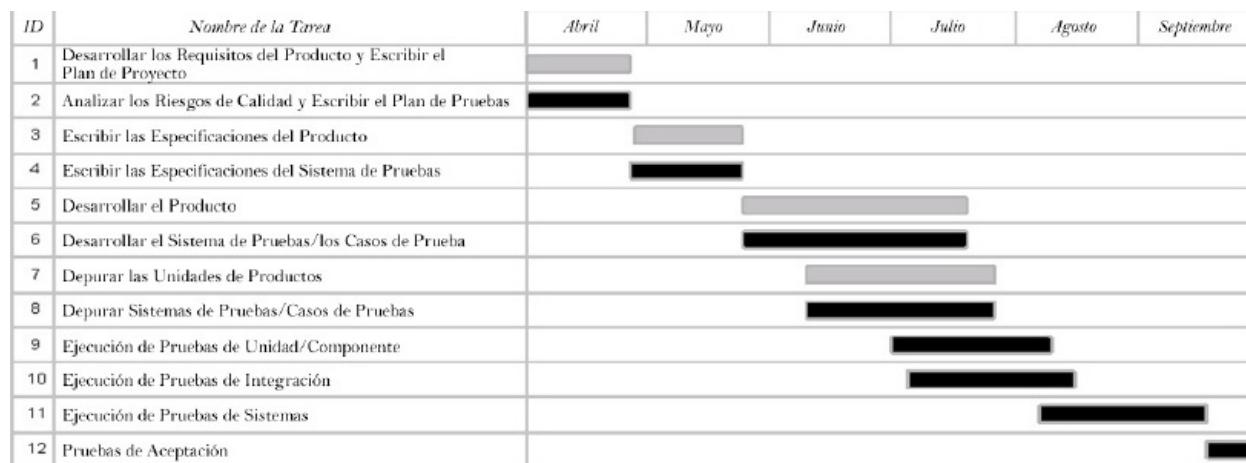


Figura 2.9: Cuando las Pruebas se Extienden por los Proyectos

En esta figura, aparece un diagrama de Gantt o una vista de la cronología de un proyecto donde se usan las pruebas dominantes. Las barras grises indican las tareas de desarrollo, mientras que las barras negras indican las tareas de pruebas.

Tenga en cuenta que las tareas de pruebas se producen en todo el trabajo del desarrollo. Hay una tarea de pruebas paralela para cada tarea de desarrollo, que corresponde a nuestra regla anterior que decía que cada producto del trabajo debería ser probado.

Como puede ver, los períodos de ejecución de las pruebas unitarias, de integración y de sistema son más largos que las pruebas de aceptación. Eso se debe en parte al hecho, de que cuando se

realiza apropiadamente, la ejecución de las pruebas es planificada con múltiples ciclos para permitir tiempo de reparación. Como hemos mencionado antes, una de las peores prácticas es planificar, diseñar o ejecutar las pruebas suponiendo que esas pruebas pasarán y no revelarán ningún defecto.

Finalmente, aunque no es visible en esta figura, las buenas pruebas dominantes requieren el constante balance de las características, el presupuesto, el cronograma y las compensaciones de calidad. Entonces — y esto es especialmente importante durante los proyectos de ciclo de vida secuencial — las características deberían ser evaluadas acerca de la posible omisión en lugar de cambiar las fechas de entrada a los niveles de pruebas o entrar en un nivel de pruebas antes de que el objeto de pruebas esté listo para ello. El cambio de las fechas de entrada de las pruebas cuando las fechas de entrega están fijas, simplemente lo conduce a realizar la clasificación de las pruebas, reduciendo la cobertura y aumentando el riesgo de la calidad. Comenzando un nivel de pruebas en un objeto que no está listo para probar, es usualmente ineficiente y frustrante.

En los ciclos de vida iterativos, en lugar de abandonar una función, lo mejor es moverla en una iteración posterior. Esto podría tener el efecto de abandonarla de la versión, por supuesto, si esa iteración no es llevada a cabo realmente o si el producto es publicado antes de la iteración que está siendo realizada.

En esta nota acerca de los modelos del ciclo de vida, este gráfico supone un modelo secuencial del ciclo de vida. Las pruebas dominantes son bien útiles sólo en los modelos iterativos del ciclo de vida, pero era más fácil para nosotros ilustrar los puntos que estábamos realizando en esta figura utilizando un modelo secuencial.

2.2.1 Ejercicios

Ejercicio 1

Lea el Documento de Requisitos del Sistema Omninet.

Si estuviera dirigiendo todas las pruebas de Omninet, ¿Cuáles niveles o fases de pruebas planificaría? ¿Por qué?

¿Cuáles serían las metas más importantes de cada nivel o fase de pruebas?

¿Cuál tipo de pruebas de aceptación, si alguna, planificaría? ¿Por qué?

¿Cómo se relacionan y afectan estos niveles de pruebas al modelo del ciclo de vida que ha seleccionado en el ejercicio previo?

Argumente.

Solución del Ejercicio 1

Planificaríamos todos los cuatro niveles.

Nos aseguraríamos de que las pruebas de unidad (de componente) ocurrieran para todas las piezas que componen el quiosco, el centro de llamadas y el centro de datos. Quisiéramos ver tantos defectos eliminados rápidamente y económica mente como sea posible. La utilización de arneses de pruebas automatizados como J-Unit¹⁶ o J-Test¹⁷ figuraría en nuestro método para este nivel de pruebas.

Nos aseguraríamos de que haya ocurrido una rigurosa prueba de integración. Quiero encontrar tantos defectos de integración tan temprano como sea posible, y validar la arquitectura total del sistema, incluyendo en condiciones de estrés, carga y error. Planificaríamos una prueba de integración que procedería a través de una secuencia de adiciones cuidadosas y pequeñas. Esperaríamos que ocurriera unas seis o siete construcciones. Utilizaríamos un método de red central donde examinamos las comunicaciones a través de tres subsistemas principales, también como parte de las pruebas de integración.

Una equivocación clásica de pruebas en las pruebas de integración de un sistema como Omninet sería probar el centro de llamadas, el centro de datos y el quiosco por ellos mismos y luego encontrar que ellos no funcionan juntos una vez integrados.

Con una sólida prueba de componente e integración realizada, incluyendo las pruebas de integración de toda la funcionalidad que abarca los componentes, la prueba de sistema, mientras es rigurosa, no debería encontrar demasiados defectos. Mayormente nos enfocaríamos en los defectos de las acciones de punta a punta, el rendimiento de todo el sistema y la respuesta a la carga y la funcionalidad esencial.

También planificaríamos una prueba de aceptación, sin embargo la llamaríamos una prueba beta. Utilizaríamos clientes reales en sitios seleccionados para realizar esta prueba. Esta prueba beta sería acerca de la búsqueda de los defectos más que la mejora de la confianza del usuario en el sistema.

Estos cuatro niveles coinciden con el modelo V, que seleccionamos en el ejercicio anterior. Sin embargo, estos desarrollos para usuarios finales y mercados masivos, las pruebas beta ocurren en paralelo con las pruebas de sistema—o podrían empezar incluso durante las pruebas de

integración.

2.3 Tipos u Objetivos de las Pruebas

Objetivos del Aprendizaje

LO-2.3.1 Comparar los cuatro tipos de pruebas de software (funcionales, no funcionales, estructurales y relacionadas con el cambio) por medio de un ejemplo. (K2)

LO-2.3.2 Reconocer que las pruebas funcionales y estructurales ocurren en cualquier nivel de las pruebas. (K1)

LO-2.3.3 Identificar y describir los tipos de pruebas no funcionales basados en los requisitos no funcionales. (K2)

LO-2.3.4 Identificar y describir los tipos de pruebas basados en el análisis de una estructura o arquitectura de un sistema de software. (K2)

LO-2.3.5 Describir el propósito de las pruebas de confirmación y las pruebas de regresión. (K2)

Esta sección, Tipos u Objetivos de las Pruebas, cubrirá los siguientes conceptos clave:

- Principales tipos u objetivos de las pruebas de software.
- Pruebas funcionales y no funcionales.
- Pruebas estructurales.
- Pruebas de confirmación y regresión.
- Utilización de las pruebas funcionales, no funcionales y estructurales en varios niveles.

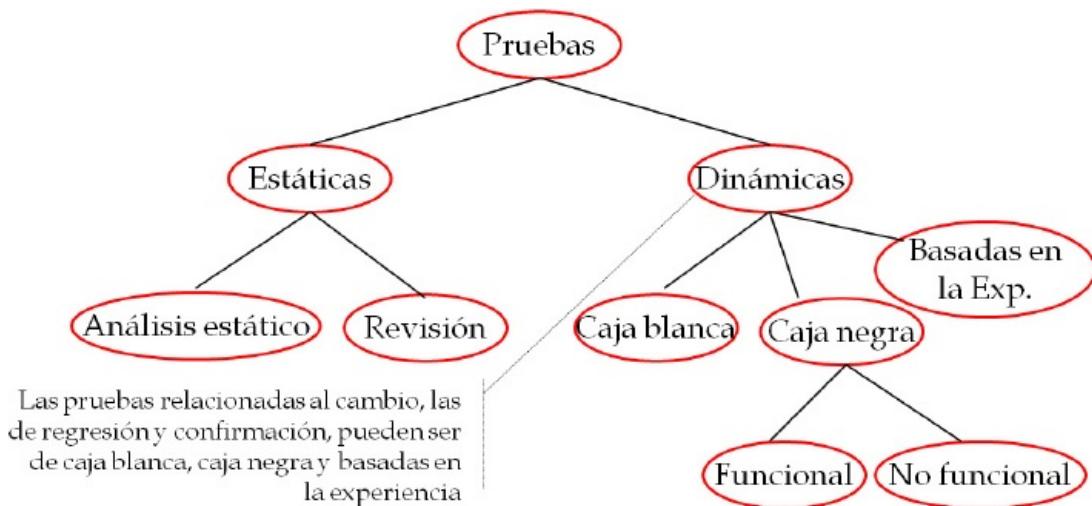


Figura 2.10: Comparación de Los Tipos de Pruebas

El programa de estudios básico del ISTQB define una taxonomía de los tipos de pruebas, como se puede ver en esta figura.

Existen tipos de pruebas estáticas, las cuales son cualquier tipo de pruebas que no involucran la ejecución del ítem sometido a pruebas. Los tipos de pruebas estáticas consisten en el análisis estático—el implica la utilización de una herramienta para evaluar el ítem sometido a pruebas—y las revisiones—las cuales involucran la evaluación manual del ítem sometido a pruebas. Cubriremos las pruebas estáticas en el capítulo 3.

Existen tipos de pruebas dinámicas, las cuales son cualquier tipo de pruebas que involucran la ejecución del ítem sometido a pruebas. Cubriremos las pruebas dinámicas en el capítulo 4.

Un subtipo de las pruebas dinámicas es la prueba estructural o de caja blanca. El diseño de las pruebas estructurales está basado principalmente en la estructura del ítem sometido a prueba. Otro subtipo de pruebas dinámicas es la prueba de comportamiento o de caja negra. El diseño de pruebas de comportamiento se basa principalmente en el comportamiento del ítem sometido a prueba.

Glosario del ISTQB

Pruebas de caja negra: Pruebas, ya sea funcionales o no funcionales, sin referencia a la estructura interna del componente o sistema.

Pruebas basadas en la especificación: Véase pruebas de caja negra.

Otro subtipo de las pruebas dinámicas son las pruebas basadas la experiencia. El diseño de las pruebas basas en la experiencia se basa en el conocimiento y la experiencia del probador, los cuales pueden abarcar tanto los aspectos de comportamiento como los estructurales del ítem sometido a prueba.

Los tipos de pruebas de comportamiento se dividen además en pruebas funcionales y pruebas no funcionales. En términos generales, las pruebas funcionales prueban el “qué” del comportamiento del ítem, mientras que las pruebas no funcionales prueban el “cómo” del comportamiento del ítem. El programa de estudios básico del ISTQB utiliza el estándar ISO 9126 para clasificar las pruebas

de comportamiento ya sean funcionales o no funcionales. Introduciremos el estándar ISO 9126 al final de esta sección.

El programa de estudios básico también presenta una taxonomía alternativa relacionada con las pruebas dinámicas. Esta taxonomía examina la utilización de una prueba para evaluar los resultados de un cambio en el software existente.

Un tipo de prueba relacionado con el cambio es la prueba de confirmación, la cual es llamada también repetición de pruebas. Las pruebas de confirmación, naturalmente suficientes, consisten en la confirmación de que el cambio fue implementado correctamente. Alternativamente el otro tipo de prueba relacionado con el cambio es la prueba de regresión. Las pruebas de regresión comprueban si el cambio afecta negativamente a otras partes del sistema.

Revisemos más de cerca a estos tipos de pruebas.

Vamos a empezar con las pruebas funcionales. Las pruebas funcionales buscan situaciones donde la acción razonable o necesaria es proporcionada o no, inaccesible o seriamente deteriorada. Por ejemplo, si no hay ninguna función de adición en una calculadora, podemos decir razonablemente que a la calculadora le falta una función esencial. Si la función de adición está implementada, pero la tecla “+” no funciona para invocarla, la función está básicamente inaccesible. Si la función de adición sólo puede sumar números enteros, no números reales, está tan seriamente deteriorada para ser inútil.

Las pruebas funcionales deberían, por supuesto, verificar que el resultado de una acción sea correcto. Si la función de adición suma números, pero calcula que $2+2=5$, eso es claramente un defecto. Para verificar la funcionalidad correcta, necesitamos lo que se llama un oráculo de pruebas. Un oráculo de pruebas es cualquier fuente de información que nos permite determinar los resultados esperados para comparar con el resultado real del software sometido a pruebas. Un oráculo puede ser el sistema existente (para una evaluación comparativa), un manual de usuarios o el conocimiento especializado de una persona, pero no debería ser el código. (Si utiliza el código como un oráculo, estará sólo comprobando que el compilador y enlazador funcionan.) Si usted tiene una especificación para la funcionalidad del sistema, subsistema, componente o de cualquier ítem que está sometido a pruebas— si este documento es una especificación de requisitos, casos de uso o especificación funcional — entonces ése es sin duda un oráculo de pruebas. Sin embargo, a menudo algunas funciones permanecen indocumentadas y los probadores deben entender “el comportamiento razonable”. Por supuesto, es posible también que la especificación esté errónea y los probadores deben tomar en cuenta esto.

Las pruebas funcionales pueden también comprobar los efectos secundarios e indeseables. Por ejemplo, si la función de dividir funciona en una calculadora, pero cambia el formato de la calculadora a un formato de números romanos, eso sería un defecto.

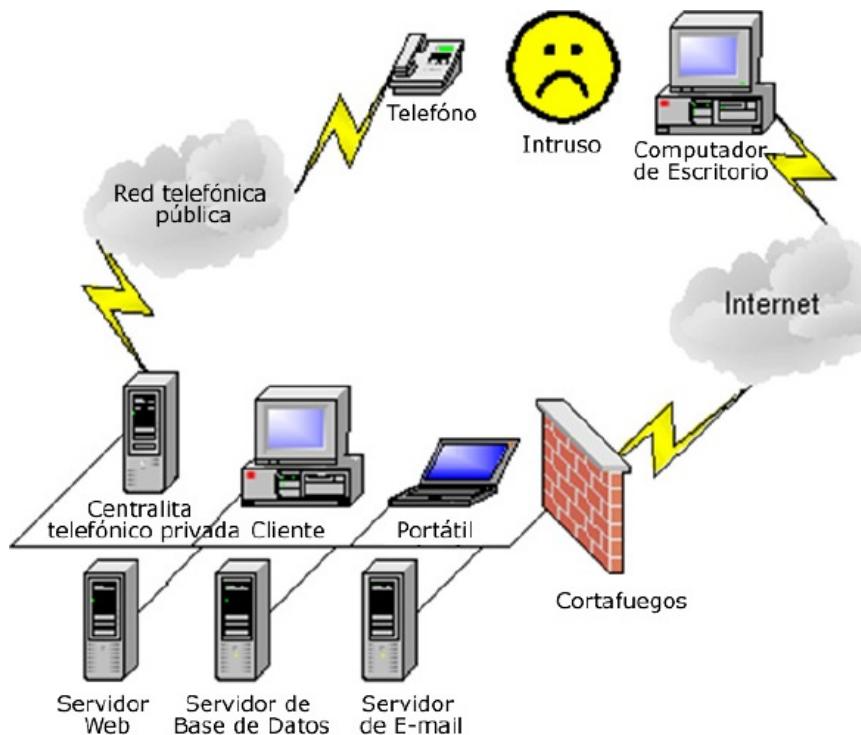


Figura 2.11: Seguridad

Un tipo especializado de las pruebas funcionales son las pruebas de seguridad, generalmente realizadas por expertos en pruebas especiales de seguridad.

Las amenazas de seguridad incluyen los ataques de los virus, los Caballos Troyanos, los Botnets18

y otros ataques a través de la oficina del usuario, también incluyen el intento de entrar en un servidor directamente. (Los servidores — y en particular los datos que residen en ellos — son generalmente los objetivos finales). Los ataques de la denegación del servicio y la denegación distribuida del servicio tienen por objetivo hacer a los servidores inaccesibles por los usuarios legítimos ya sea violando la seguridad del servidor o sobrecargándolo.

Existen muchas expectativas y suposiciones equivocadas entre las personas que pertenecen al área de seguridad. Por ejemplo, algunas personas piensan que sólo utilizando la encriptación (es decir, HTTPS, SSL y etc.) en el servidor Web se resuelven los problemas de seguridad. Si los datos son almacenados sin encriptación, entonces son vulnerables.

Algunas personas piensan que con sólo la compra e instalación de un cortafuego (“firewall”) se resuelven todos los problemas seguridad. Los cortafuegos deben ser configurados por alguien que sabe lo que ellos hacen para funcionar correctamente.

Algunas organizaciones confían demasiado en los administradores de redes o sistemas no calificados para identificar y resolver los problemas de seguridad. Una vez hicimos una prueba de seguridad para un cliente como parte de un proyecto de desarrollo de un sistema de internet. Más tarde, antes de su publicación, el cliente nos pidió ejecutar la prueba de nuevo. Todos los problemas que se habían identificado en los servidores durante la primera ronda fueron resueltos — para esos servidores. Sin embargo, para los nuevos servidores que se habían añadido todos los problemas anteriores se presentaron de nuevo.

Otro tipo especializado de las pruebas funcionales son las pruebas de interoperabilidad. La mayoría de las aplicaciones tienen que correr con otras aplicaciones en el mismo entorno, así que tenemos que comprobar que el objeto de pruebas — el ítem sometido a pruebas — pueda funcionar correctamente con estos programas, con el sistema operativo o los sistemas, que lo hospedarán, con las bases de datos que éste interactúa o cohabita, y así sucesivamente. La interoperabilidad se enfoca en las interfaces directas entre los ítems que interactúan, pero también en las vías indirectas del intercambio de datos. Podemos realizar pruebas de interoperabilidad de pares (“pairwise testing”)¹⁹, una-por-una y podemos hacer pruebas de interoperabilidad de punta a punta en colecciones de ítems.

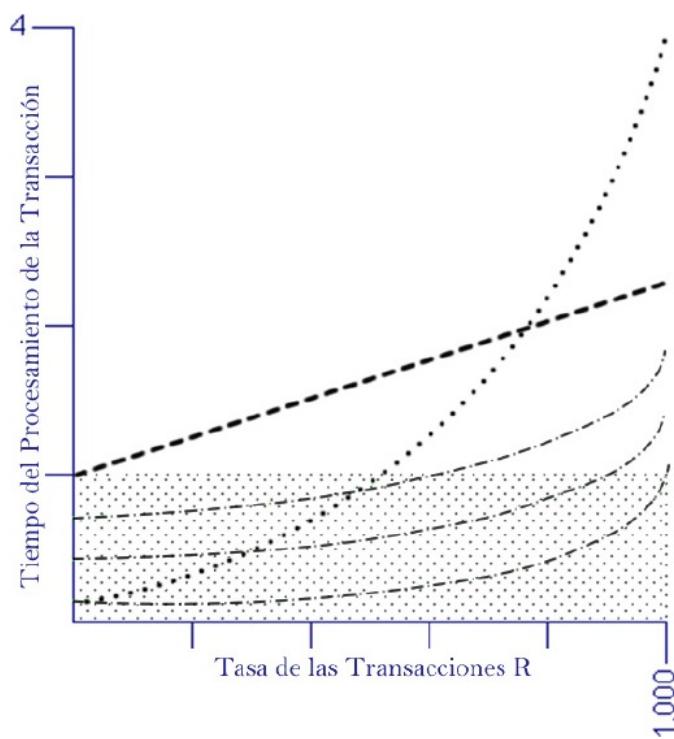


Figura 2.12: Rendimiento y Fiabilidad

Pasando a las pruebas no funcionales, empecemos con las pruebas de rendimiento y fiabilidad.

Supongamos que estamos probando un sistema que se supone que debe ser capaz de procesar una transacción — por ejemplo, una consulta de un cliente acerca de si un artículo está disponible — en un segundo, en cargas de hasta 1.000 transacciones por minuto. El rectángulo sombreado muestra el área de rendimiento aceptable.

En el gráfico de esta figura, vemos tres tipos comunes de los problemas del rendimiento.

El primero, ilustrado por la línea discontinua gruesa, es donde el sistema responde demasiado lento a lo largo de la curva del rendimiento, sin importar el nivel de carga.

El segundo, ilustrado por la línea punteada, es donde el sistema tiene una “elevación” inaceptable en la curva del rendimiento. En alrededor de 500 transacciones por minuto, el tiempo de respuesta

se hace inaceptablemente lento y la degradación es claramente no lineal bajo carga a partir de alrededor de 250 transacciones por minuto.

El tercer problema, ilustrado por el grupo de las líneas discontinuas delgadas, es donde el rendimiento del sistema es aceptable cuando se prueba por primera vez. Sin embargo, la repetición de pruebas posteriores sin reiniciar el servidor muestra degradaciones del rendimiento inaceptables con el tiempo.

Este tercer problema se relaciona a menudo con un problema de fiabilidad. Las pruebas de fiabilidad buscan situaciones en las que el sistema falla al completar funciones normales, legales, ya sea debido a las restricciones en la secuencia o algún otro defecto y donde el sistema funciona normalmente, pero se bloquea o cuelga aleatoriamente.

Las pruebas de rendimiento y fiabilidad a menudo involucran la creación de condiciones de pruebas de estrés, la utilización intensa de la capacidad y altos volúmenes de carga. Las pruebas de estrés implican la creación de condiciones extremas para causar fallas, a menudo a través de una combinación de las pruebas del forzamiento del error, pruebas de capacidad y volumen. Las pruebas de capacidad buscan los problemas de funcionalidad, rendimiento o fiabilidad, debido al agotamiento de los recursos, así como llenar un disco duro o memoria a más del 80% utilizable. Las pruebas de volumen buscan también los problemas de funcionalidad, rendimiento o fiabilidad, pero en este caso debido a la tasa de los flujos de datos, así como la ejecución de más del 80% de las transacciones calculadas por minuto, más del 80% de usuarios concurrentes, etc.

Glosario del ISTQB

Pruebas de estrés: Un tipo de pruebas de rendimiento conducido para evaluar un sistema o componente en o fuera de los límites de sus cargas de trabajo anticipadas o especificadas o con menor disponibilidad de los recursos así como el acceso a la memoria o los servidores. Véase también las pruebas de rendimiento y pruebas de carga.

Generalmente, una vez que la utilización de un recurso alcanza el 80%, comenzará una degradación funcional, de rendimiento y fiabilidad.

Continuando con los tipos de pruebas no funcionales, las pruebas de mantenimiento y mantenibilidad pueden ser también importantes.

Glosario del ISTQB

Pruebas de mantenibilidad: El proceso de pruebas para determinar la mantenibilidad de un producto de software.

Mantenibilidad: La facilidad con la cual un producto de software puede ser modificado para corregir defectos, modificado para satisfacer nuevos requisitos, modificado para hacer el futuro mantenimiento más fácil o ser adaptado a un entorno modificado. Tenga en cuenta que este término no ha sido enunciado específicamente en esta sección, pero se lo incluye aquí, porque es esencial para la comprensión del término pruebas de mantenibilidad.

Las pruebas de mantenimiento incluyen la comprobación de los problemas con los procesos de actualización e instalación y desinstalación de parches. También incluyen la comprobación de los problemas con los cambios de configuración legal, así como el conectar y listo ("plug-and-play"), la adición de espacio de disco, y así sucesivamente.

Las pruebas de mantenibilidad incluyen la comprobación de que si el propio software (código fuente) no es mantenible, o si la base de datos no puede ser actualizada con una nueva versión. Otro problema de mantenibilidad sería cuando una base de datos crece monótonamente — es decir, sin parar —, si ese crecimiento excedería eventualmente la capacidad del almacenamiento o tal vez más grave, causaría problemas de rendimiento. Las cuestiones de comprobabilidad y regresión están también relacionadas con el mantenimiento.

Otras pruebas no funcionales incluyen las pruebas de usabilidad e interfaz de usuario. Un sistema puede funcionar correctamente pero puede ser inutilizable por el cliente meta o el usuario. Esto puede pasar cuando el sistema presenta interfaces pesadas que no siguen los flujos del trabajo. O cuando el sistema es inapropiadamente difícil de comprender para los usuarios. O cuando éste presenta mensajes instructivos, de ayuda y de error los cuales son erróneos, confusos o con errores ortográficos.

Las pruebas de configuración y portabilidad son también no funcionales. Si una sola plataforma de hardware se puede configurar de diferentes maneras por el software, o si un grupo de plataformas puede ser compatible con varias configuraciones de hardware, entonces se necesitan pruebas para esas variaciones. En algunos casos, las configuraciones pueden cambiar y eso debería ser probado. Por ejemplo, si añadimos espacio de disco u otro dispositivo de almacenamiento, añadimos memoria y actualizamos con una nueva versión o añadimos potencia al CPU. Si una aplicación se va ejecutar con múltiples sistemas operativos, bases de datos, necesitamos probar la portabilidad en estos entornos.

Como si esta lista no fuera ya lo suficientemente larga, hay todavía otras pruebas que podrían ser consideradas como pruebas de comportamiento funcional y no funcional, tales como:

- Localización (interfaz de usuario).
- Localización (operacional).
- Estándares y cumplimiento regulatorio.
- Control de errores y recuperación.

- Recuperación de desastres.
- En red/internet o distribuido.
- Sincronización y coordinación.
- Calidad de datos.
- Conversión de datos.
- Operaciones.
- Instalación.
- Desinstalación.
- Control de fecha y hora.
- Documentación.
- Y muchos otros...

Adicionalmente a las pruebas de comportamiento tenemos las pruebas estructurales. Éstas son pruebas que se basan en cómo el sistema está construido, ya sea en el código, en su estructura de datos o en su diseño.

En algunos casos, hemos medido la cobertura basada en la estructura (de caja blanca) de las pruebas de comportamiento funcional y no funcional para comprobar las omisiones. Estos análisis de cobertura pueden ser alarmantes con muy poco de la estructura real normalmente cubierta. Este tema se tratará con mayor profundidad más adelante...

Como hemos mencionado anteriormente, también podemos clasificar las pruebas como pruebas de regresión o confirmación.

Las pruebas de regresión comprueban los efectos de los cambios en las partes no cambiadas del sistema. Cambios pequeños, localizados y aislados no siempre tienen efectos pequeños, localizados o aislados. Cubriremos las estrategias de regresión en la siguiente sección.

Las pruebas de confirmación están para confirmar que los cambios introducidos en el sistema estén presentes y que funcionan correctamente, o para confirmar que las correcciones de los defectos introducidas en el sistema resuelven los síntomas observados, es decir, las fallas previamente informadas.

Es útil tener pruebas repetibles durante las pruebas de regresión y confirmación. De otra manera, no podríamos comprobar las mismas condiciones en nuestras pruebas y podríamos perder los defectos de regresión o los problemas con las correcciones.

La automatización, cubierta en profundidad más adelante, es también útil para las pruebas de regresión y confirmación.

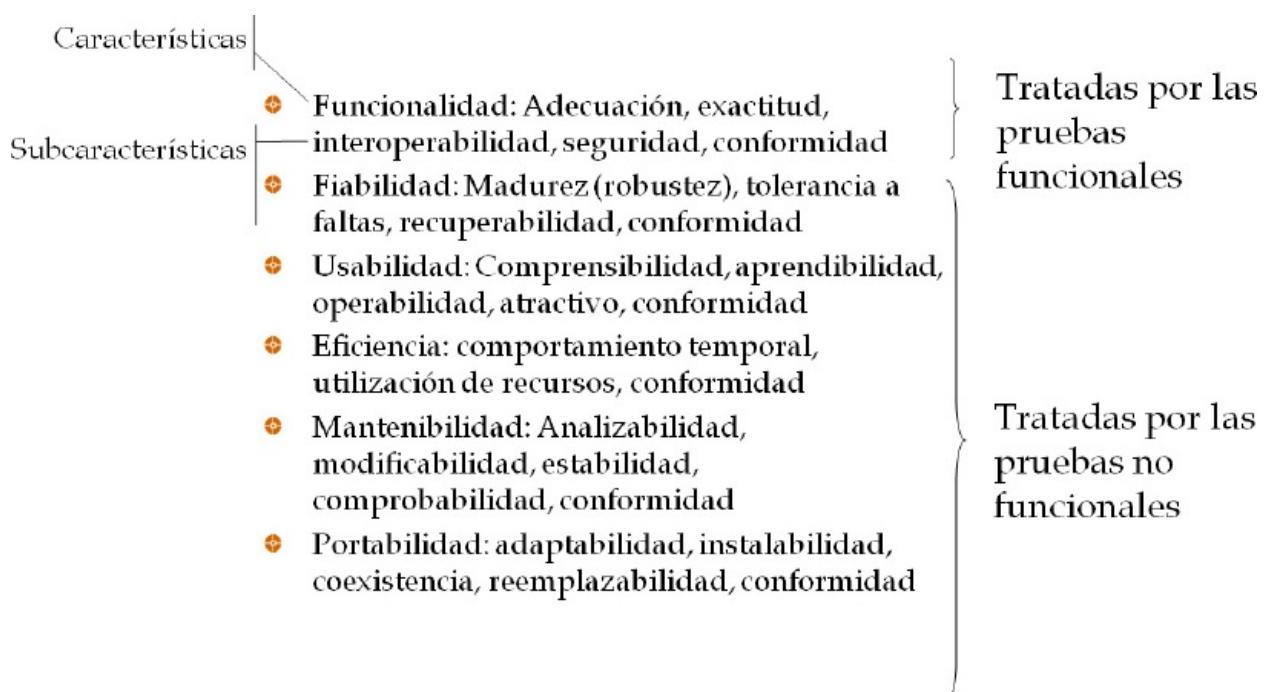


Figura 2.13: El Estándar de Calidad ISO 9126

Mencionamos el estándar ISO 9126 antes. Para finalizar esta sección, permítanos presentarle este estándar en detalle.

El estándar de calidad para el software ISO 9126 define seis características de calidad del software: Funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Cada característica tiene tres o más subcaracterísticas, como se muestra en esta figura.

Las pruebas que abordan la funcionalidad y sus subcaracterísticas son las pruebas funcionales.

Las pruebas que abordan a las otras cinco características y sus subcaracterísticas son las pruebas no funcionales.

Le recomendamos que estudie esta figura por un momento, porque, así como con los estándares CMMI y IEEE 12207, se espera que usted sea capaz de recordar esto en el examen.

2.3.1 Ejercicios

Ejercicio 1

Refiriéndonos nuevamente a los niveles que usted ha seleccionado en el ejercicio anterior, haga una lista de los objetivos o los tipos de pruebas que incluiría en cada nivel.

¿Cómo afecta el modelo del ciclo de vida que seleccionó en el ejercicio anterior a la cantidad de las pruebas de regresión?

¿Cómo afecta el modelo del ciclo de vida que seleccionó en el ejercicio anterior a la cantidad de las pruebas de confirmación?

Argumente.

Solución del Ejercicio 1

En los niveles de pruebas de integración y componentes, incluiríamos la funcionalidad y el rendimiento de cada componente y a través de los componentes. También planificaríamos las pruebas de usabilidad de las interfaces del usuario del quiosco y el centro de llamadas. Temprano pondríamos los prototipos de las interfaces del usuario en frente de los clientes y agentes reales del centro de llamadas tan rápido como sea posible.

El nivel de pruebas de sistema se enfocaría en las clases de problemas que sólo podríamos encontrar en el sistema integrado y completo. Planificaríamos pruebas de sistema de los siguientes tipos:

- Funcionalidad, particularmente las funciones de punta a punta como la interacción del usuario del quiosco, con los agentes del centro de llamadas.
- Seguridad en cuanto a los programadores perfeccionistas y obsesivos (“hackers”) penetrando el centro de datos, en el centro de llamadas o los quioscos y las potenciales brechas de seguridad, involuntarias e intencionadas por los clientes del quiosco o agentes del centro de llamadas.
- Rendimiento en niveles anticipados de carga, volumen y capacidad, así como también las situaciones realistas y de estrés.
- Fiabilidad, especialmente para medir el tiempo promedio entre la falla de los quioscos y la recuperación sin técnicos de reparación teniendo que visitar los quioscos.
- Mantenimiento y mantenibilidad en cuanto a la capacidad de instalar varios tipos de actualizaciones y parches para el quiosco, el centro de llamadas y los sistemas del centro de datos.
- Usabilidad y la interfaz de usuario, los cuales serían especialmente importantes, sin embargo con la esperanza de que la mayoría de esos problemas hayan sido encontrados antes del comienzo de la prueba de sistema.
- Configuración, en cuanto a los diferentes tipos de quioscos compatibles y la portabilidad de los sitios Web importantes a través de estas configuraciones diferentes.
- Localización, para las zonas de horarios, los eventos activados por el tiempo y los idiomas.

En las pruebas beta, le pediríamos a los usuarios del quiosco y los agentes del centro de llamadas que consideren la funcionalidad en cuanto a los posibles problemas con los que ha sido provista así como también las funciones que deberían haber sido provistas pero que no lo fueron. También quisiéramos la retroalimentación del usuario acerca de la usabilidad y el rendimiento. Planificaríamos también la introducción de carga en el centro de datos para obtener la comprobación de los usuarios de la usabilidad y el rendimiento en ambas condiciones normales y de uso intenso.

Comparado con los modelos incrementales, el modelo V reduce las pruebas de regresión. Esto es porque usted no comienza la prueba de sistema hasta que todos los componentes estén realizados, lo cual reduce la cantidad de rotación del para el sistema después de que la prueba de sistema empieza. Mientras menos cambios más bajo el nivel de los riesgos de regresión. Sin embargo el modelo V, no altera dramáticamente las pruebas de confirmación. La cantidad de pruebas de confirmación es dirigida por el número y tipo de defectos encontrados.

2.4 Pruebas de Mantenimiento

Objetivos del Aprendizaje

LO-2.4.1 Comparar las pruebas de mantenimiento (las pruebas de un sistema existente) con las pruebas de una nueva aplicación con respecto a los tipos de pruebas, los activadores para las pruebas y la cantidad de las pruebas. (K2)

LO-2.4.2 Reconocer los indicadores para las pruebas de mantenimiento (modificación, migración y retiro). (K1)

LO-2.4.3. Describir el rol de las pruebas de regresión y el análisis del impacto en el mantenimiento. (K2)

Glosario del ISTQB

Análisis del impacto: La evaluación del cambio en las capas de la documentación del desarrollo, la documentación de las pruebas y los componentes, con el fin de implementar un cambio dado en los requisitos especificados.

Pruebas de mantenimiento: Pruebas de los cambios en un sistema en producción o el impacto de un entorno modificado en un sistema en producción.

Esta sección, Pruebas de Mantenimiento, cubrirá los siguientes conceptos clave:

- Razones para las pruebas de mantenimiento.
- Pruebas de mantenimiento versus pruebas de aplicaciones nuevas.
- El rol de las pruebas de regresión y el análisis del impacto.

Hay una variedad de razones por las que los sistemas se someten al mantenimiento — y por lo tanto las pruebas de mantenimiento:

La más obvia es la modificación, como ser las mejoras, las correcciones de defectos, los cambios del entorno operativo y los parches.

Otro desencadenante común para el mantenimiento es la migración, trasladándose a un nuevo entorno compatible.

Tal vez menos común es el retiro, cuando el fin de la vida útil de un subsistema o todo el sistema desencadena en un reemplazo. Los datos del antiguo sistema deben ser archivados con frecuencia de una manera que nos permita utilizarlos, si no fueron totalmente migrados al nuevo sistema.

Las pruebas de mantenimiento deberían abordar tanto el cambio en sí mismo, así como también lo que no fue modificado y lo que no debería ser modificado.



Figura 2.14: Pruebas de las Versiones de Mantenimiento

Esta figura muestra una vista gráfica de un diagrama de Gantt o una cronología del proyecto de mantenimiento de dos meses y medio. Al igual que una figura anterior en este capítulo, que mostró cómo las pruebas dominantes se extienden por todo el ciclo de vida del desarrollo, esta figura muestra las pruebas dominantes extendiéndose en el proyecto de mantenimiento. Como antes, las tareas del desarrollo son mostradas en gris y las tareas de las pruebas en negro. Las pruebas unitarias no se muestran por separado, sino que ocurren dentro de la tarea de “desarrollar las correcciones y las características”.

En las pruebas de las versiones de mantenimiento, hay algunos desafíos que deben ser considerados.

En primer lugar, en cuanto al impacto potencial, la regresión es el gran riesgo para las versiones de mantenimiento. La mayoría de los usuarios y clientes se ponen bastante molestos cuando usted arruina algo que estaba funcionando. Por ejemplo, Apple lanzó una actualización del software para el iPod de Rex Black, unas tres semanas después de haberlo comprado y él lo instaló tontamente, pensando: “Sin duda, ya había sido probado adecuadamente, al menos con los modelos más nuevos como el de -él”. Incorrecto. El iPod ahora se cuelga aleatoriamente donde deja de responder a la mayoría de las entradas. Su impresión acerca de la calidad de los productos de Apple bajó completamente por esa experiencia.

Otro reto es el problema “diez libras de material en un cubo de cinco libras”, donde las organizaciones tratan de poner una versión importante llena de características en una versión corta de mantenimiento. Este problema tiende a empeorar cuando los interesados del negocio dispares influyen en el contenido de una versión de mantenimiento, o cuando las versiones de mantenimiento salen con poca frecuencia, o simplemente cuando la gerencia no comprende o no aceptará las restricciones del mantenimiento y las pruebas de mantenimiento.

Dado que la atención se centra a menudo en la regresión, los equipos de pruebas tienen a veces problemas para encontrar el tiempo y los recursos suficientes para desarrollar nuevas pruebas para nueva funcionalidad. Esto es especialmente cierto cuando los grandes juegos de pruebas de regresión automatizados deben ser preparados para la versión de mantenimiento.

Por último, las reglas heurísticas de la estimación de las pruebas en proyectos grandes fracasará a menudo para las versiones de mantenimiento. Para el desarrollo nuevo, tal vez uno puede utilizar reglas heurísticas acerca de la proporción aproximada del trabajo del desarrollo respecto al trabajo de las pruebas, así como la regla antigua, "Un probador por cada tres desarrolladores". Nosotros no avalamos este método para la estimación de las pruebas en ninguna circunstancia, y con seguridad no para las versiones de mantenimiento. Con las versiones de mantenimiento, el alcance de las pruebas es fuertemente influenciado por la cantidad necesaria de las pruebas de regresión, y esa es una función del tamaño del conjunto de las características del sistema existente, no el número de desarrolladores que trabajan en la versión de mantenimiento.

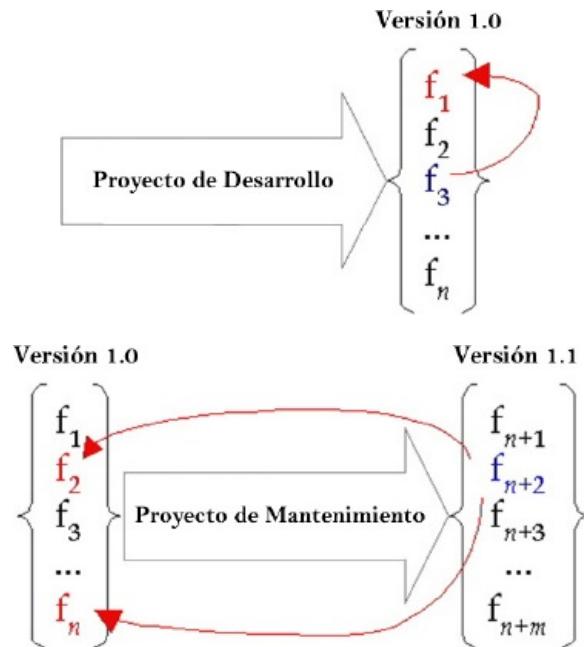


Figura 2.15: Regresión

Entonces, hablemos acerca de la regresión y las estrategias de las pruebas de regresión.

La regresión es el nombre dado al comportamiento de un sistema que ha retrocedido, o se ha degradado, en comparación con su estado actual o pasado, debido a un cambio. Se origina de los defectos introducidos o descubiertos en las zonas no modificadas del software, como un resultado de los cambios realizados.

Podría parecer extraño que haciendo un cambio, se pueda introducir un defecto en un área **no modificado**, pero esto es sólo debido a la inadecuada metáfora del mundo físico que a menudo nos hacen pensar acerca del software. El software a diferencia de los objetos físicos de tres dimensiones como los puentes o los automóviles, está lleno de interconexiones, suposiciones entrecruzadas, caminos de datos y otras maneras en las que un componente afecta, directa o indirectamente, a otro componente, en algunos casos un componente sin ninguna relación obvia con el componente modificado.

La regresión puede ser clasificada en dos formas útiles. La primera en términos de la relación de la distancia funcional entre el cambio y el defecto de regresión que éste introduce. En esta clasificación, podemos referirnos a las regresiones locales, donde el cambio crea un nuevo defecto en la misma área funcional, el cual no debería ser demasiado difícil de encontrar.

Podemos referirnos a una regresión expuesta, donde el cambio revela defectos existentes en la misma área funcional, los cuales podrían ser un poco más difíciles de encontrar. También podemos referirnos a una regresión a distancia, donde un cambio en un área daña algo en otra área. Dependiendo de cuán "lejos" funcionalmente hablando, es el cambio del defecto de regresión, podría ser difícil encontrar este tipo de problemas.

También podemos clasificar la regresión basada en que si la característica, la función o el atributo que fueron dañados, han sido lanzados o no.

En la regresión de una característica nueva, un cambio en una nueva característica que está siendo desarrollada para esta versión — característica 3 en este gráfico — daña otra característica que está siendo desarrollada para esta versión -característica 1 en este gráfico. Ahora, eso es decepcionante, pero al menos usted no dañó nada de lo que ya estaba en uso.

En la regresión de las características existentes, un cambio en o la introducción de una nueva

característica que está siendo desarrollada para esta versión—la característica $n+2$ en este gráfico—daña una o más características que fueron previamente publicadas—las características 2 y n en este gráfico. Aquí, hemos dañado algo que ya está en uso. Si las características rotas son muy importantes, los procesos de negocio críticos podrían ser afectados.

Entonces, ¿Qué podemos hacer acerca de la regresión? Hay dos estrategias de pruebas para hacer frente a la regresión y también otras tres estrategias que pueden reducir el riesgo de la regresión.



Figura 2.16: Estrategia de Regresión 1: Repetir Todas las Pruebas

Para la primera estrategia de pruebas, tenemos la opción de la fuerza bruta. Cualquier momento que algo cambia, repita cada prueba que alguna vez ejecutó. Aquí, si nuestras pruebas abordan los riesgos de calidad importantes, entonces repitiendo todas las pruebas debería encontrar las regresiones más importantes.

Como algo práctico, la automatización es la única forma de hacer esto para sistemas complejos y grandes.

Cubriremos la automatización más a fondo en una sección posterior, pero vale la pena mencionar una fábula Checa antigua en el contexto de esta opción.

En esta leyenda Checa, un cierto rabino de Praga creó una figura humana de barro y le dio vida. El rabino intento utilizar esta cosa, llamada golem, para llevar a cabo ciertas tareas útiles para él mismo y sus feligreses. Sin embargo en un momento debido a alguna desgracia u otra, el golem se enloqueció e hizo bastante daño antes de ser sometido por el rabino.

Ésta es sin duda una metáfora apropiada para las pruebas totalmente automatizadas. Un montón de gente, en su intento de la implementación de juegos de pruebas de regresión completamente automatizados, ha creado un golem que se enloqueció.

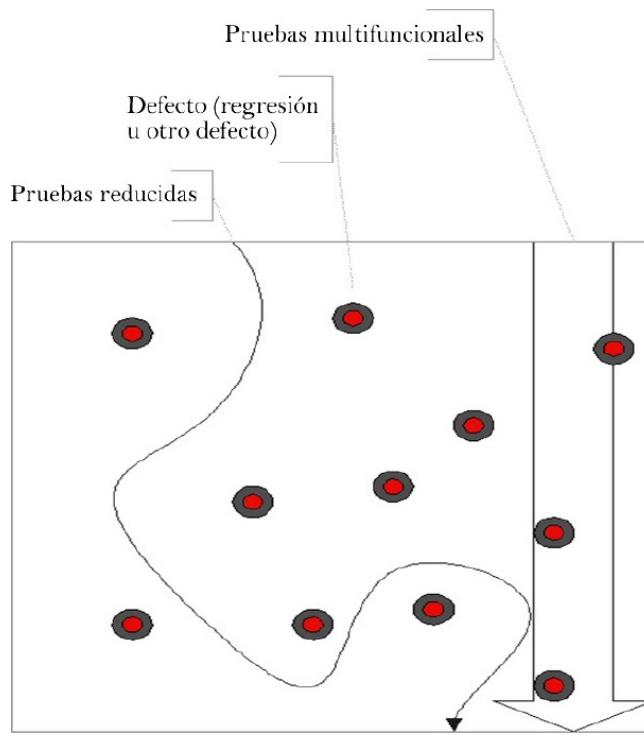


Figura 2.17: Estrategia de Regresión 2: Repetir Algunas Pruebas

La segunda estrategia de pruebas es repetir algunas pruebas pero no todas. Si la automatización completa no es posible, ésta es la estrategia que usted terminará utilizándola.

Por supuesto, en la medida que la automatización sea práctica para algunas pruebas, realícela por todos los medios, pero, para las áreas del sistema no cubiertas por las pruebas automatizadas, tendrá que seleccionar las pruebas manuales que deben ser repetidas. Idealmente, las pruebas son seleccionadas basadas en la probabilidad de descubrir un defecto de regresión o en el impacto que un defecto tendría si éste existiese.

Comenzando con la probabilidad, tenemos dos herramientas para utilizarlas en esto. Podemos utilizar la trazabilidad de nuestras pruebas hacia los requisitos y el diseño para tratar de comprender cómo el cambio podría crear la regresión. (Cubriremos la trazabilidad más tarde si no está familiarizado con el concepto.) Básicamente, volvemos a ejecutar cualquier prueba que se relaciona con los requisitos o ítems del diseño afectados por el cambio.

Podemos utilizar el análisis del cambio o el impacto, donde miramos directamente al cambio para tratar de anticipar a sus efectos secundarios. Normalmente esto se hace preguntando al programador que realizó el cambio. Por supuesto, el hecho de que estamos confiando en el programador es una debilidad de esta técnica, ¡esas son las relaciones acerca de las cuales él **no pensó** que el más probablemente las dañe!

Por último, en cuanto al impacto potencial, nos referiremos a nuestro análisis de los riesgos de calidad. (De nuevo, este tema será tratado más adelante). Si hemos identificado ciertas áreas que simplemente no pueden ser dañadas, por temor a las graves consecuencias para el negocio, los usuarios, etc., entonces esas áreas—ya sea que probablemente sean dañadas por el cambio o no—deberían ser probadas de nuevo.

Para tratar de aumentar la eficiencia de estas pruebas de regresión manuales, podemos crear pruebas extensas, multifuncionales que abarcan áreas amplias del sistema. Esto resulta en lo que a veces llamamos “pruebas de regresión accidentales”, en el sentido de que tenemos pruebas de un “día típico de oficina” que cubren muchas áreas, y por lo tanto, están bien posicionadas para encontrarse con defectos.

Si usted está realmente preocupado acerca de cuántos riesgos de regresión quedan, una técnica consiste en utilizar la cobertura del código para evaluar el nivel de riesgo restante después de haber ejecutado las pruebas de regresión.

El concepto de cobertura de código se explicará más adelante, pero, por el momento, piense de esta manera: Mientras un defecto se podría esconder en el código que ha probado, es mucho más fácil para el defecto de esconderse en el código que **no ha** probado. La cobertura de código comprueba para ver cuánto del código no ha probado y además cuántos lugares hay, donde se esconden los defectos de regresión.

Glosario del ISTQB

Cobertura de código: Un método del análisis que determina cuáles partes del software han sido ejecutadas (cubiertas) por el juego de pruebas y cuáles partes no han sido ejecutadas, p.ej., la cobertura de sentencias, la cobertura de decisiones o la cobertura de condiciones.

Hay otras tres estrategias las cuales son estrategias más de gestión de proyectos que estrategias de pruebas, pero que sin embargo ayudan a reducir el riesgo de la regresión.

La primera es publicar la versión más lentamente. Si usted publica una versión cada seis meses en lugar de cada mes, usted debería ser capaz de volver a ejecutar un conjunto más grande de sus pruebas si está utilizando una estrategia manual. Incluso podría tener más tiempo para dedicarlo a la automatización, resultando en la repetición completa.

La segunda, si debe haber parches de emergencia entre las versiones más lentas, combine esos parches en las versiones más grandes y más lentas. Esto permite algo de flexibilidad de las versiones más frecuentes mientras se mantiene bajo el riesgo de regresión total, sobre todo para los que no necesitan instalar los parches.

La tercera, involucrar a los clientes o usuarios en las pruebas (por supuesto con la venia del personal del proyecto y la gerencia de negocios). En el mercado público, esto se llama pruebas beta, mientras que en el mundo de TI se llaman versiones piloto, de etapas o fases o paralelas. De cualquier manera, si hay regresiones pero ninguno de sus probadores piloto o beta las encuentran, deberían ser menos importantes que las que ellos hubiesen encontrado si estuviesen allí.

2.4.1 Ejercicios

Ejercicio 1

Asuma que puede automatizar completamente las pruebas del quiosco de Omnidet y el centro de llamadas. Si un cambio es realizado después de la versión en la interfaz del usuario del centro de llamadas que no afecta la funcionalidad, ¿Qué probaría de nuevo?

Asuma que no ha automatizado las pruebas del quiosco de Omnidet y el centro de llamadas. Si el mismo cambio es realizado, ¿Qué probaría de nuevo? ¿Qué si el cambio si afectó la funcionalidad?

Argumente.

Solución del Ejercicio 1

Si tiene un conjunto completamente automatizado de pruebas para el centro de llamadas, el método de riesgo mínimo, desde un punto de vista de la calidad, es probar de nuevo todo el centro de llamadas después de cada cambio. Sin embargo, ¿Supone usted que toma cuatro semanas para ejecutar todo el conjunto de pruebas automatizadas? La demora asociada con la finalización de las pruebas podría conducirlo a ejecutar sólo las pruebas de la interfaz del usuario. Esto es especialmente verdadero si tiene solamente las pruebas manuales que no sólo consumen tiempo sino que también son costosas en cuanto al esfuerzo humano.

La calidad de la arquitectura del sistema también influye la decisión. Si sabe que el centro de llamadas es una aplicación web donde la interfaz del usuario vive completamente en el navegador, mientras toda la lógica de negocios vive en los servidores, entonces la realización de sólo las pruebas de la interfaz del usuario tiene más sentido. Si mucho de la lógica de negocios está en el cliente, entonces cualquier cambio al cliente representa algún riesgo a la funcionalidad del sistema.

Preguntas de Examen de Muestra y Simulación

Para finalizar cada capítulo, usted puede tratar de resolver una o más preguntas de examen de muestra para reforzar su conocimiento y comprensión del material y prepararse para el examen del Probador ISTQB Nivel Básico.

Sección 2.1 Modelos de desarrollo de software (K2)

#	Pregunta	K
34.	<p>Objetivo del aprendizaje: LO-2.1.1</p> <p>Usted está trabajando como el único probador en un proyecto pequeño con un modelo V, que acaba de publicar un borrador de la especificación de los requisitos. Considere las siguientes posibilidades:</p> <ol style="list-style-type: none">I. Usted debería participar en una revisión del borrador.II. Usted debería utilizar el borrador de la especificación de los requisitos para empezar el análisis y diseño de las pruebas de aceptación.III. El borrador de la especificación de los requisitos puede servir como una base de las pruebas.IV. El borrador de la especificación de los requisitos puede servir como una base inmodificable de las pruebas. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ol style="list-style-type: none">A. I, II, III, y IV son todas verdaderas.B. II, III, y IV son verdaderas.	2

	C. Sólo la I es verdadera. D. I, II, y III son verdaderas.	
35.	Objetivo del aprendizaje: LO-2.1.2 ¿Cuál de las siguientes afirmaciones es verdadera acerca de la adaptación de los modelos de desarrollo de software? A. Usted puede adaptar los modelos para que se acomoden a las varias características de los proyectos y productos. B. No debería proceder así, porque los que originaron los modelos son expertos. C. No debería adaptar los modelos basados en los resultados de las pruebas, para evitar comprometer la calidad. D. No debería preocuparse por esos modelos, porque no afectan las pruebas.	1
36.	Objetivo del aprendizaje: LO-2.1.3 ¿Cuál de las siguientes es una característica de buenas pruebas en cualquier proyecto, en cualquier nivel de prueba, independientemente del modelo del ciclo de vida? A. El objetivo principal de las pruebas es encontrar tantos defectos como sea posible. B. Las pruebas de regresión no son afectadas por el modelo del ciclo de vida. C. Los probadores deberían participar en la revisión de documentos. D. Los probadores no deberían saber acerca de los detalles estructurales de los productos que ellos están probando.	1
37.	Objetivo del aprendizaje: término ¿Qué es la validación? A. La confirmación de que los requisitos especificados se han cumplido. B. La confirmación de que los requisitos para una utilización o aplicación intencionada y específica han sido cumplidos. C. El proceso de probar un sistema integrado para verificar que cumple los requisitos especificados. D. La parte de la gestión de la calidad enfocada en proporcionar la confianza en que los requisitos serán cumplidos.	1
38.	Objetivo del aprendizaje: Estándar CMMI. Considere los siguientes niveles de madurez del CMMI: I. Inicial. II. Optimizado. III. Definido. IV. Gestionado. V. Gestionado cuantitativamente. ¿Cuál afirmación pone estos niveles en su orden correcto? A. I, II, III, IV, V. B. I, III, IV, V, II. C. I, IV, III, V, II. D. II, I, III, IV, V.	1
39.	Objetivo del aprendizaje: Estándar IEEE 12207. ¿Cuál de los siguientes se aborda como una sección principal en el estándar IEEE 12207? A. Los principales procesos del ciclo de vida. B. Los criterios de suspensión/reanudación. C. Las revisiones técnicas. D. Los refinamientos del Método.	1

Sección 2.2 Niveles de pruebas (K2)

#	Pregunta	K
40.	Objetivo del aprendizaje: LO-2.2.1 Las pruebas de las interfaces entre componentes son un principal objetivo ¿de cuál nivel de pruebas?	2

	A. Prueba de componente. B. Prueba de integración. C. Prueba de sistema. D. Prueba de aceptación.	
41.	Objetivo del aprendizaje: término. ¿Qué son las pruebas operacionales? A. El proceso de probar para determinar el rendimiento de un producto del software. B. Las pruebas realizadas para evaluar un componente o un sistema en su entorno operacional. C. El proceso de probar para determinar la recuperabilidad de un producto del software. D. Las pruebas que ejecutan los casos de prueba que fallaron la última vez que fueron ejecutados.	1

Sección 2.3: Tipos de pruebas (K2)

Estándares

- [ISO 9126] ISO/IEC 9126-1: 2001, Ingeniería de software - Calidad del producto de software].

Términos

Pruebas de caja negra, cobertura de código, pruebas funcionales, pruebas de interoperabilidad, pruebas de carga, pruebas de mantenimiento, pruebas de rendimiento, pruebas de portabilidad, pruebas de fiabilidad, pruebas de seguridad, pruebas basadas en la especificación, pruebas de estrés, pruebas estructurales, pruebas de usabilidad y pruebas de caja blanca.

#	Pregunta	K
42.	Objetivo del aprendizaje: LO-2.3.1 Considere los siguientes cuatro tipos de pruebas para una aplicación de comercio electrónico: I. Las pruebas de cada posible consulta de la base de datos. II. Las pruebas de los tiempos de respuesta del sistema bajo carga. III. Las pruebas de nuevas características a medida que son adicionadas al sistema. IV. La verificación del manejo correcto de los pedidos típicos. ¿Cuál de las siguientes afirmaciones es verdadera? A. I. es una prueba relacionada con el cambio; II es una prueba no funcional; III es una prueba funcional y IV es una prueba estructural. B. I es una prueba estructural; II es una prueba no funcional; III es una prueba relacionada con el cambio y IV es una prueba funcional. C. Las cuatro son pruebas funcionales. D. I es una prueba no funcional; II es una prueba estructural; III es una prueba relacionada con el cambio y IV es una prueba funcional.	2
43.	Objetivo del aprendizaje: LO-2.3.2 ¿Cuál de las siguientes afirmaciones es verdadera acerca de las pruebas de funcionalidad y de caja negra? A. Nunca son ejecutadas por los programadores. B. Nunca son útiles durante las pruebas de componente. C. Estos diseños de pruebas son siempre controlados activamente mientras son realizados. D. Ellas pueden ser útiles por todos los probadores durante cualquier nivel de pruebas.	1
44.	Objetivo del aprendizaje: LO-2.3.3 Usted está probando una nueva aplicación de software para el mercado masivo y quiere asegurarse de que ésta gustará a sus usuarios meta. Usted ensambla una muestra representativa de los usuarios potenciales para un estudio y para hacer que ellos traten de lograr las tareas típicas basadas en los prototipos de las pantallas. Usted observa cuidadosamente la habilidad de ellos para terminar estas tareas sin que ellos se confundan o atasquen. ¿Cuál tipo de prueba está usted realizando? A. Pruebas de estrés. B. Pruebas de usabilidad. C. Pruebas funcionales. D. Pruebas de seguridad.	2

<p>45. Objetivo del aprendizaje: LO-2.3.4</p> <p>Usted está probando un sistema bancario web que permitirá a los clientes acceder a sus cuentas en Internet. Usted recibe un borrador de la especificación del diseño técnico que describe la arquitectura del sistema a al nivel de componentes de software. Tan pronto como usted recibe este documento, comienza a diseñar las pruebas para asegurar que cada interacción posible entre los componentes de hardware y software, ambos de sistema a sistema y de punta a punta, sea ejercida durante el proceso de los niveles de pruebas de integración y sistema. ¿Cuál tipo de prueba está usted diseñando?</p> <ul style="list-style-type: none"> A. Prueba estructural. B. Prueba de rendimiento. C. Prueba de portabilidad. D. Prueba funcional. 	<p>2</p>
<p>46. Objetivo del aprendizaje: LO-2.3.5</p> <p>¿Cuáles son los propósitos de las pruebas de confirmación y las pruebas regresión?</p> <ul style="list-style-type: none"> A. Las pruebas de regresión verifican el éxito de las acciones correctivas; las pruebas de confirmación aseguran que no han sido introducidos o descubiertos defectos en áreas no modificadas del software, como resultado de un cambio. B. Las pruebas de regresión y pruebas confirmación son sinónimos; ambas verifican el éxito de las acciones correctivas. C. Las pruebas de confirmación verifican el éxito de las acciones correctivas; las pruebas de regresión aseguran que no han sido introducidos o descubiertos defectos en áreas no modificadas, como resultado de un cambio. D. Las pruebas de regresión y pruebas confirmación son sinónimos; ambas aseguran que no han sido introducidos o descubiertos defectos en áreas no modificadas, como resultado de un cambio. 	<p>2</p>
<p>47. Objetivo del aprendizaje: Estándar ISO 9126</p> <p>De acuerdo al estándar ISO 9126, ¿cuál de las siguientes es una característica de calidad?</p> <ul style="list-style-type: none"> A. La asignación de recursos. B. El tiempo de respuesta. C. La seguridad. D. La eficiencia. 	<p>1</p>
<p>48. Objetivo del aprendizaje: término</p> <p>¿Qué son las pruebas de caja negra?</p> <ul style="list-style-type: none"> A. La derivación o selección de los casos de prueba basados en un análisis de la especificación del sistema o las componentes. B. La derivación o selección de los casos de prueba basados en un análisis de la estructura interna del sistema o componente. C. El proceso de la identificación de las diferencias entre los resultados reales producidos por el componente o sistema sometido a pruebas y los resultados esperados para una prueba. D. Una técnica informal del diseño de pruebas, donde el probador controla activamente el diseño de las pruebas a medida que esas pruebas son realizadas. 	<p>1</p>

Sección 2.4: Pruebas de mantenimiento (K2)

Términos

Análisis del impacto y pruebas de mantenimiento.

#	Pregunta	K
<p>49. Objetivo del aprendizaje: LO-2.4.1</p> <p>¿Cuál tipo de prueba es típicamente más importante durante las pruebas de mantenimiento en comparación a las pruebas de una aplicación completamente nueva?</p> <ul style="list-style-type: none"> A. Pruebas de regresión. B. Pruebas de rendimiento. C. Pruebas funcionales. D. Pruebas de seguridad. 	<p>2</p>	

50.	<p>Objetivo del aprendizaje: LO-2.4.2</p> <p>Considere las siguientes situaciones:</p> <ul style="list-style-type: none"> I. Una función existente en el sistema de producción está siendo modificada para hospedar nuevos clientes. II. Un sistema cliente servidor es migrado a un nuevo servidor huésped, el cual utiliza un CPU diferente que el servidor huésped actual. III. Los datos del fin de la vida útil de un sistema están siendo archivados en una cinta para un posible análisis futuro. IV. Un pedido de cambio ha sido aprobado en la especificación de la línea base durante un proyecto de desarrollo. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Cualquiera de estas situaciones podrían dar lugar a las pruebas de mantenimiento. B. Sólo II y IV darían lugar a las pruebas de mantenimiento. C. Sólo I, II, y III darían lugar a las pruebas de mantenimiento. D. Sólo I y II darían lugar a las pruebas de mantenimiento. 	1
51.	<p>Objetivo del aprendizaje: LO-2.4.3</p> <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Durante las pruebas de mantenimiento, el análisis del impacto nos ayudó a decidir cuántas pruebas de regresión deben ser realizadas. B. Durante las pruebas de mantenimiento, el análisis del impacto nos ayudó a decidir cuántas pruebas de confirmación deben ser realizadas. C. Durante las pruebas de nuevo desarrollo y mantenimiento, el análisis del impacto nos ayuda a decidir cuántas pruebas de confirmación deben ser realizadas. D. El análisis del impacto no afecta a las pruebas de regresión. 	2
52.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué son las pruebas de mantenimiento?</p> <ul style="list-style-type: none"> A. Las pruebas de los cambios en un sistema operacional o el impacto del cambio del entorno de un sistema operacional. B. Las pruebas de un programa previamente probado después de la modificación para asegurar que no han sido introducidos o descubiertos defectos en áreas no modificadas del software. C. Las pruebas que ejecutan los casos de prueba que fallaron la última vez cuando ellos fueron ejecutados, para verificar el éxito de las acciones correctivas. D. Las pruebas basadas en el análisis de la especificación de la funcionalidad de un componente o sistema. 	1

Capítulo 2 Preguntas a través de las secciones

#	Pregunta	K
53.	<p>Cubre: sección 2.1 y 2.2.</p> <p>¿En cuál modelo del ciclo de vida es más probable que el nivel de pruebas de integración ocurra paralelamente durante una parte significativa del nivel de prueba de sistema?</p> <ul style="list-style-type: none"> A. Iterativo. B. Modelo V. C. De regresión. D. De rendimiento. 	2

Preguntas del Examen de Simulación 1

#	Pregunta	K
54.	<p>Objetivo del aprendizaje: LO-2.1.1</p> <p>¿Cuál de los siguientes es un modelo del ciclo de vida de desarrollo abordado en el programa de estudios básico del ISQTB?</p> <ul style="list-style-type: none"> A. Iterativo. B. Verificación. C. Validación. D. CMM. 	1

55.	<p>Objetivo del aprendizaje: LO-2.3.1 También utiliza las secciones 2.2 y 2.3 Considere los siguientes productos del trabajo, los tipos de pruebas y los niveles de pruebas:</p> <ul style="list-style-type: none"> I. El documento del diseño de la arquitectura del sistema. II. El documento de los casos de uso. III. Las pruebas funcionales. IV. Las pruebas estructurales. V. Las pruebas de integración. VI. Las pruebas de aceptación. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. II es útil para IV durante V; I es útil para III durante VI. B. I es útil para IV durante V; II es útil para III durante VI. C. II es útil para IV durante V; pero I no es útil para III durante VI. D. I no es útil para IV durante V; pero II es útil para III durante VI. 	2
56.	<p>Objetivo del aprendizaje: LO-2.2.1 ¿Cuál de los siguientes es un nivel de pruebas abordado en el programa de estudios básico del ISQTB?</p> <ul style="list-style-type: none"> A. Prueba de componente. B. Prueba de regresión. C. Prueba de confirmación. D. Prueba de rendimiento. 	1
57.	<p>Objetivo del aprendizaje: LO-2.3.1 Usted está probando el tiempo de respuesta de un sistema bajo carga durante una prueba de integración. ¿Cuál tipo de prueba está usted realizando?</p> <ul style="list-style-type: none"> A. Funcional. B. No- funcional. C. Estructural. D. Relacionada al cambio. 	2
58.	<p>Objetivo del aprendizaje: término. ¿Qué es un modelo del ciclo de vida iterativo?</p> <ul style="list-style-type: none"> A. Un ciclo de vida de desarrollo donde un proyecto es dividido en una serie de iteraciones. B. Un ciclo de vida de desarrollo donde un proyecto es tercerizado a una serie de equipos de desarrollo. C. Un ciclo de vida de desarrollo donde un proyecto tiene múltiples niveles de pruebas, los cuales son considerados iteraciones. D. Un ciclo de vida de desarrollo donde el código es escrito por dos programadores sentados en una sola computadora. 	1
59.	<p>Objetivo del aprendizaje: LO-2.4.3 ¿Qué es un análisis del impacto?</p> <ul style="list-style-type: none"> A. Las pruebas de un sistema previamente probado después de la modificación. B. El proceso de evaluación de los riesgos identificados para estimar su impacto y probabilidad de ocurrencia. C. La evaluación del impacto de un cambio en las capas de la documentación del desarrollo, la documentación de las pruebas y los componentes. D. Las pruebas de los cambios en un sistema operacional o el impacto de la modificación del entorno del sistema operacional. 	1

Preguntas del Examen de Simulación 2

#	Pregunta	K
60.	<p>Objetivo del aprendizaje: LO-2.1.2 Si durante un proyecto, siguiendo el modelo del ciclo de vida iterativo, el jefe del proyecto decide publicar un sistema sin las últimas características del incremento debido a un gasto inesperado del presupuesto del proyecto, ¿Qué es lo que él ha hecho?</p> <ul style="list-style-type: none"> A. Priorizó las restricciones presupuestarias más alto que la calidad. B. Tomó una decisión que el Jefe de pruebas no debería aprobar. C. Adaptó el ciclo de vida del desarrollo al proyecto. 	1

	D. Cambió un ciclo de vida del modelo V	
61.	<p>Objetivo del aprendizaje: LO-2.4.3 Considere lo siguiente:</p> <ul style="list-style-type: none"> I. Análisis del impacto. II. Pruebas de confirmación. III. Pruebas de mantenimiento. IV. Pruebas de regresión. V. Diseño del sistema. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. I es útil para II durante III. B. I es útil para IV durante V. C. I es útil para IV durante III. D. II es útil para I durante III. 	2
62.	<p>Objetivo del aprendizaje: LO-2.3.3 Un producto de software de un sistema operativo para el mercado masivo diseñado para que sea ejecutado en cualquier hardware de computador personal con un procesador de la familia x86. Usted está ejecutando un conjunto de pruebas para buscar los defectos relacionados con la compatibilidad de varios computadores personales que utilizan ese procesador y para construir la confianza en que las marcas importantes de los computadores personales funcionarán. ¿Cuál tipo de prueba está usted realizando?</p> <ul style="list-style-type: none"> A. Prueba de rendimiento. B. Prueba de procesador. C. Prueba funcional. D. Prueba de interoperabilidad. 	2
63.	<p>Objetivo del aprendizaje: término ¿Qué son las pruebas operacionales?</p> <ul style="list-style-type: none"> A. Las pruebas de los cambios en un sistema operacional o el impacto de la modificación del entorno de un sistema operacional. B. Las pruebas realizadas para evaluar un componente o sistema en su entorno operacional. C. Las pruebas de aceptación por los usuarios o clientes en su sitio, normalmente incluyendo el hardware así como el software. D. El proceso de pruebas para determinar la mantenibilidad de un producto de software. 	1
64.	<p>Objetivo del aprendizaje: LO-2.2.1 ¿En cuál tipo de defectos un nivel de componentes podría centrar la identificación?</p> <ul style="list-style-type: none"> A. Los defectos en los módulos u objetos que son comprobables separadamente. B. Los defectos en interfaces entre los componentes o interacciones con diferentes partes de un sistema. C. Los defectos en todo el sistema o producto. D. Los niveles de pruebas de componente no se enfocan en la identificación de defectos. 	1
65.	<p>Cubre: sección 2.2 y 2.3 Usted está ejecutando una prueba de rendimiento bajo una carga pesada con el objetivo de encontrar los cuellos de botella en la comunicación de la red que ocurre a través de las interfaces entre cuatro de los 16 componentes de un sistema. ¿Qué está haciendo usted posiblemente?</p> <ul style="list-style-type: none"> A. Una prueba funcional durante el nivel de pruebas de integración. B. Una prueba no funcional durante el nivel de pruebas de integración. C. Una prueba funcional durante el nivel de pruebas de componente. D. Una prueba no funcional durante el nivel de pruebas de componente. 	1

12 Un applet es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.

13 En este contexto se habla de quiosco como una tienda pública de acceso a internet no asistida. En algunos países se los llama café internet en otros cibercafé y etc.

14 Real Academia Española: Gran explosión en que una teoría cosmogónica sitúa el origen del universo.

- 15 Pruebas dominantes son aquellas pruebas que consiguen las personas idóneas para trabajar juntas a través del proceso correcto en el tiempo correcto para un alto Retorno de Inversión. A través de las pruebas dominantes, todas las ideas que hemos explorado hasta ahora se unen.
- 16 JUnit es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.
- 17 Un paquete de calidad completo de los desarrolladores de java para el análisis estático de código (de seguridad, fiabilidad y rendimiento), revisión de código y pruebas de unidad o componente.
- 18 Botnet es un término que hace referencia a un conjunto de robots informáticos o bots, que se ejecutan de manera autónoma y automática.
- 19 Pruebas de pares (o de todos los pares) es una técnica efectiva de generación de casos de prueba que se basa en la observación de que la mayoría de los defectos son causados por las interacciones de máximo dos factores. Los juegos de pruebas generados por pares cubren todas las combinaciones de los pares en un conjunto más pequeño que todas las pruebas posibles pero aún muy efectivo en el hallazgo de los defectos.

Capítulo 3

Técnicas Estáticas

"Nada ocurre porque si. Todo en la vida es una sucesión de hechos que, bajo la lupa del análisis, responden perfectamente a causa y efecto"

Richard Feynmann, premio nobel de física acerca de la electrodinámica cuántica.

El capítulo 3, Técnicas Estáticas, contiene las siguientes 3 secciones:

1. Técnicas estáticas y el proceso de pruebas.
2. Proceso de revisión.
3. Análisis estático por medio de herramientas.

Cada una de las secciones estará desglosada en dos o más partes.

3.1 Técnicas Estáticas y el Proceso de Pruebas

Objetivos del Aprendizaje

LO-3.1.1 Reconocer los productos del trabajo del software que pueden ser examinados por las diferentes técnicas estáticas. (K1)

LO-3.1.2 Describir la importancia y el valor de considerar las técnicas estáticas para la evaluación de los productos del trabajo del software. (K2)

LO-3.1.3 Explicar las diferencias entre las técnicas estáticas y dinámicas, considerando los objetivos, los tipos de defectos que deben ser identificados y el rol de estas técnicas en el ciclo de vida del software. (K2)

Esta sección, Técnicas Estáticas y el Proceso de Pruebas, cubrirá los siguientes conceptos clave:

- Productos del trabajo del software y técnicas estáticas.
- Importancia y valor de las técnicas estáticas.
- Diferencia entre las técnicas estáticas y dinámicas.
- Objetivos del análisis estático y las revisiones y la comparación del objetivo con las pruebas dinámicas.

Las pruebas estáticas son aquellas pruebas donde el ítem sometido a pruebas—el objeto de prueba — no tiene que ser ejecutado (o debe funcionar).

Glosario del ISTQB

Pruebas dinámicas: Pruebas que involucran la ejecución del software de un componente o sistema.

Pruebas estáticas: Pruebas de un componente o sistema en un nivel de la especificación o implementación sin la ejecución de ese software, p.ej. las revisiones o el análisis estático.

Análisis estático: Análisis de los artefactos del software, p.ej., los requisitos o el código llevados a cabo sin la ejecución de estos artefactos del desarrollo de software. El análisis estático es usualmente llevado a cabo por una herramienta.

Esto puede involucrar la utilización de las revisiones y las herramientas. Ahora, las revisiones, las cuales pueden ir de informales a muy formales, serán cubiertas en detalle en breve. Hay también varias herramientas que pueden realizar algunos tipos de pruebas estáticas, así como la comprobación de la conformidad con los estándares de codificación. Se pueden utilizar las pruebas estáticas para los requisitos y los diseños, lo cual es bastante típico. También se pueden—y se deberían— utilizar para el código, los esquemas de las bases de datos, la documentación, las pruebas y cualquier otro producto de trabajo que haya sido escrito.

Una forma de pruebas estáticas es la utilización de modelos y prototipos. Un diagrama de un sistema complejo puede revelar con frecuencia problemas de diseño que pueden esconderse en las palabras. Un diagrama de diseño deformé significa a menudo muchos defectos.

En un proyecto de hace algunos años, utilizamos las pruebas estáticas de los modelos del diseño para comprobar si habían problemas potenciales de rendimiento. Las pruebas estáticas incluían un modelo descrito en una hoja de cálculo acerca de la utilización de los recursos en distintos niveles de carga. También incluían un modelo de simulación ejecutable de la utilización de los recursos del sistema. Esto previno una gran cantidad de defectos que de otro modo se habrían descubierto durante la prueba de sistema.

Usted también debería considerar a la creación de casos de prueba y datos de prueba como una forma de pruebas estáticas.

El análisis y el diseño de las pruebas basados en las especificaciones de los requisitos y el diseño son una forma de revisión estructurada, y estos procesos de análisis y diseño de pruebas revelan a menudo el problema. Este ejemplo de las pruebas es considerado como una actividad de la prevención de defectos.

Hablando acerca de las herramientas para las pruebas estáticas, tenemos diversas formas del análisis estático que podemos realizar. Una de estas formas es la comprobación de la ortografía, la gramática y la dificultad de lectura que se pueden realizar contra los documentos. Por ejemplo, Word incluye la capacidad para detectar los errores de ortografía, los problemas de gramática y los pasajes de lectura difíciles, los cuales pueden ser útiles para las especificaciones de los requisitos y el diseño.

Al observar el código fuente, podemos utilizar herramientas para comprobar los constructos peligrosos de programación. Estas herramientas son por ejemplo J-Test, Safer C, lint y otras. También podemos utilizar las herramientas de código abierto y las comerciales para medir el código en cuanto a los asuntos del análisis de la complejidad, la cual será explicada más adelante en este libro.

Glosario del ISTQB

Complejidad: El grado en que un componente o sistema tiene un diseño y/o una estructura interna que es difícil de entender, mantener y verificar. Véase también la complejidad ciclomática.

Las herramientas de análisis estático también contienen las simulaciones de sistemas. Hay un programa llamado General Purpose System Simulator, el cual ha estado en uso durante décadas que puede simular componentes básicos de un sistema, como las colas y los recursos. Como se mencionó antes, existen herramientas de modelado de rendimiento e investigación de operaciones que pueden predecir el comportamiento del sistema sometido a carga. Incluso podemos utilizar herramientas simples como las hojas de cálculo para simular el comportamiento del sistema, especialmente para el rendimiento, pero también para la funcionalidad.

Empecemos con la técnica manual de las revisiones. Las revisiones deberían ser utilizadas para cada producto valioso del trabajo del proyecto, en un algún nivel de formalidad u otro.

Lamentablemente, las revisiones no se aproximan a la clase de utilización y atención que se merecen. Esto es debido a la separación de los costos de las revisiones y sus beneficios.

El costo de las revisiones consiste en tres tipos principales. Primero, si vamos a tener una revisión, se necesita el tiempo necesario para realizar las revisiones. Con frecuencia, la preocupación aquí no es tanto el esfuerzo como la duración. Segundo, si estamos realizando una revisión formal, necesitamos el esfuerzo necesario para recopilar y analizar las métricas. Tercero, cuando las revisiones son utilizadas para producir valiosas métricas, esas métricas deberán ser analizadas—con algún esfuerzo—para realizar mejoras en el proceso. Estos costos no son insignificantes, pero estos producen beneficios serios más tarde en el proyecto.

Glosario del ISTQB

Revisión formal: Una revisión caracterizada por procedimientos y requisitos documentados, p.ej. La inspección.

Métrica: Una escala de medición y el método utilizado para la medición.

Estos beneficios incluyen cuatro tipos principales. El primero y quizás el más convincente, es el beneficio de cronogramas más cortos. Dado que los defectos serán eliminados antes con un menor esfuerzo (como se mostró en un capítulo anterior, debemos ahorrar tiempo). Segundo, debido a que menos defectos entrarán en las fases de pruebas, deberíamos observar períodos de pruebas más cortos y menos costosos en las pruebas. Tercero, porque el reproceso es siempre una forma de pérdida—y porque la corrección de defectos es más fácil cuando encontramos los defectos, en lugar de los síntomas, como es el caso durante las revisiones—deberíamos observar la productividad mejorada del desarrollador. Finalmente, como se mostró nuevamente en un capítulo anterior, deberíamos ver la calidad mejorada del producto, lo cual reducirá los costos indirectos, tanto en el desarrollo como después de las versiones.

En definitiva, las revisiones de todos los tipos son técnicas probadas, de alto retorno para mejorar la calidad. David Rico, en sus estudios acerca de las organizaciones del Departamento de Defensa de los Estados Unidos, encontró que las revisiones tienen un retorno de la inversión que está siempre por encima de 10-a-1; es decir, 1.000%.

Observemos ahora las similitudes y las diferencias entre las pruebas estáticas y dinámicas.

Recuerde que las pruebas estáticas son las pruebas donde el ítem sometido a pruebas—el objeto de pruebas—no es ejecutado, mientras que las pruebas dinámicas son las pruebas donde el ítem sometido a pruebas—el objeto de pruebas—es ejecutado.

Generalmente, tanto las pruebas dinámicas como las estáticas buscan identificar defectos, como uno de los principales objetivos. Ambas funcionan mejor cuando una amplia gama de interesados del negocio están involucrados. Recuerde nuestra descripción anterior acerca de las pruebas dominantes. Y también, ambas ahorran tiempo y dinero a la compañía, como fue demostrado en un capítulo anterior cuando hablamos acerca de los beneficios de las pruebas tempranas y el aseguramiento temprano de la calidad.

Entonces ¿Cuáles son las diferencias? Bueno, por un lado cada técnica puede encontrar diferentes tipos de defectos de manera más efectiva y eficiente. Por ejemplo, ciertos tipos de cuestiones de la mantenibilidad son más fáciles de encontrar en las revisiones y los análisis estáticos. Por otra

parte, las técnicas estáticas encuentran más bien defectos que fallas. En otras palabras, si realizamos una revisión y encontramos una suposición equivocada acerca del comportamiento sometido a carga, estamos examinando directamente la suposición equivocada, no el comportamiento incorrecto que ocurriría.

3.1.1 Ejercicios

Ejercicio 1

¿Considera usted las revisiones y el análisis estático útiles en el proyecto Omninet?

Si es así, ¿Cuáles tipos de problemas cree usted que estas revisiones y el análisis estático localizarían?

¿Cuáles tipos de problemas no podrían localizar?

Argumente.

Solución del Ejercicio 1

Definitivamente utilizaríamos las revisiones y el análisis estático en el proyecto Omninet. Quisiéramos ver las revisiones de los requisitos, el diseño y el código, así como también el análisis de las pruebas, los casos de prueba, el plan de pruebas y los resultados de las pruebas. Impondríamos los análisis estáticos en el código de Omninet. En el caso de Omninet, el código no solo incluye programas acerca de los servidores del centro de datos y el centro de llamadas, sino también HTML y otros entregables acerca de los clientes del centro de llamadas y los quioscos.

Las revisiones encuentran típicamente las ambigüedades, las incompletitudes y los conflictos. Los análisis estáticos encuentran código descuidado, no estándar, peligroso e inalcanzable. Sin embargo, las revisiones y el análisis estático, mientras son bastante efectivos, eliminan típicamente el 65% o menos de los defectos presentes en el ítem revisado o analizado.²⁰ Dado que cientos o incluso miles de defectos podrían existir en un sistema tan complejo como Omninet, las revisiones y los análisis estáticos deben ser ampliados con las pruebas dinámicas en los niveles de componente, integración, sistema y aceptación.

3.2 Proceso de Revisión

Objetivos del Aprendizaje

LO-3.2.1 Recordar las actividades, los roles y las responsabilidades de una revisión formal típica. (K1)

LO-3.2.2 Explicar las diferencias entre los diferentes tipos de revisión: revisión informal, revisión técnica, revisión guiada (“Walkthrough”) e inspección. (K2)

LO-3.2.3 Explicar los factores para el desempeño exitoso de las revisiones. (K2)

Glosario del ISTQB

Revisión informal: Una revisión que no se basa en un procedimiento formal (documentado).

Inspección: Un tipo de revisión por pares que se basa en la revisión visual de documentos para detectar defectos, p.ej., las violaciones de los estándares de desarrollo y la disconformidad con la documentación de nivel superior. La técnica de revisión más formal y además basada siempre en un procedimiento documentado. Véase también la revisión por pares.

Revisión técnica: Una actividad de debate de grupo por pares que se enfoca en lograr un consenso acerca del método técnico que deba tomarse. Véase también la revisión por pares.

Revisión guiada (“Walkthrough”): Una presentación paso a paso por el autor de un documento con el fin de reunir información y establecer un entendimiento común de su contenido. Véase también revisión por pares.

Esta sección, Proceso de revisión, cubrirá los siguientes conceptos clave.

- Fases, roles y responsabilidades de una revisión formal típica.
- Diferencias entre los diferentes tipos de revisiones.
- Factores para las revisiones exitosas.

Hay varios tipos de revisiones de diversos niveles de formalidad.

Hay revisiones informales. Éstas no siguen un proceso real e incluyen charlas de pasillos, pruebas de pares, programación de pares y líderes técnicos que revisan los diseños y el código. Éstas pueden tener los resultados documentados, pero el nivel de detalle en el documento es típicamente bajo. Tienen una efectividad limitada en la eliminación de los defectos, pero son útiles, económicas y populares.

La efectividad depende mucho del empleo de revisores idóneos. El propósito principal es de proporcionar una forma económica para conseguir algún beneficio.

Glosario del ISTQB

Revisor: La persona involucrada en la revisión que identifica y describe las anomalías en el producto o proyecto sometido a revisión. Los revisores pueden ser elegidos para representar los diferentes puntos de vista y los roles en el proceso de revisión.

Hay revisiones técnicas. Éstas tienen un proceso documentado y definido de la eliminación de los defectos. Debería involucrar a sus compañeros y técnicos expertos. Si el proceso no involucra jefes, entonces es una revisión entre pares. Típicamente los jefes no asistirían si no pueden contribuir técnicamente. Si el jefe puede contribuir técnicamente, entonces el proceso debería ser establecido de manera que el jefe no utilice los hallazgos de la revisión para premiar o sancionar al autor o los participantes de la revisión. Las revisiones técnicas son idealmente conducidas por un moderador entrenado quien no es el autor, aunque esto no es necesario para una revisión técnica. La preparación de una pre-reunión por los revisores es normalmente necesaria. Las listas de comprobación, las cuales son específicas para el tipo del ítem que está siendo revisado, son recomendadas pero no necesarias durante la preparación y la revisión misma. La revisión debería resultar en un informe de revisión que incluye la lista de los hallazgos, el veredicto si el producto de software cumple con sus requisitos y si es apropiado, y sus recomendaciones relacionadas con los hallazgos. En la práctica, éstas varían de bastante informal a muy formal. Los propósitos principales de una revisión técnica son el debate, la toma de decisiones, la evaluación de alternativas, el hallazgo de defectos, la solución de problemas técnicos y la comprobación de la conformidad con las especificaciones, los planes, las regulaciones y los estándares. Una revisión guiada es un tipo especial de revisión técnica donde el orden del día es el ítem sometido a la revisión. El autor guía la revisión del ítem de revisión. Esto puede incluir los escenarios, los ensayos y la participación de grupos de compañeros. Las revisiones guiadas pueden ser abiertas sin límites determinados. Tanto la preparación de la pre-reunión de los revisores como la entrega de un informe de la revisión incluyendo los hallazgos, son opcionales para las revisiones guiadas. Si un informe tiene que ser entregado, entonces debería haber un escribano o un registrador, quien no es el autor idealmente.

Como con las revisiones técnicas, en la práctica, éstas varían de bastante informal a muy formal. Los propósitos principales de una revisión guiada son aprender, ganar la comprensión y encontrar los defectos.

Glosario del ISTQB

Moderador o líder de la inspección: El líder y la persona principal responsable para una inspección u otro proceso de revisión. Tenga en cuenta que el término “líder de la inspección” no está definido en el Glosario ISTQB, pero su utilización en El Programa de Estudios del ISTQB Nivel Básico implica que es un sinónimo.

Revisión por pares: Una revisión de un producto del trabajo del software por colegas del productor del producto con el propósito de identificar los defectos y las mejoras. Ejemplos son la inspección, la revisión técnica y la revisión guiada (“Walkthrough”).

Escribano: La persona quien registra cada defecto mencionado y alguna sugerencia para la mejora del proceso durante una reunión de revisión, en un formulario de registro. El escribano debería que garantizar que el formulario de registro sea legible y comprensible.

Una inspección es el método más formal. En este método, un moderador entrenado, quien no puede ser el autor, lidera el equipo de inspección. Los compañeros son seleccionados cuidadosamente para formar el equipo de revisión. Cada miembro del equipo de revisión tiene roles definidos, basados en su experticia particular. Un proceso formal de inspección es utilizado, el cual tiene reglas, listas de comprobación, criterios de entrada y salida. El proceso incluye también la recopilación de las métricas de la eliminación de los defectos. La preparación de la pre-reunión puede ser llevada a cabo, como descrita para la revisión técnica, pero para las inspecciones esta preparación es obligatoria. Hay un proceso de seguimiento formal, incluyendo los elementosopcionales del mejoramiento del proceso. A veces se da el caso de que un lector especialmente entrenado es incorporado. El propósito principal de una inspección es de encontrar defectos—de hecho encontrar a casi todos de ellos.

Note que un método informal valora más la velocidad que la efectividad en la eliminación de los defectos, mientras que una revisión formal valora la efectividad de la eliminación de defectos. De acuerdo a los estudios de Capers Jones, las técnicas informales tienden a alcanzar una efectividad de la eliminación de los defectos cerca del 25%, mientras que las inspecciones formales pueden alcanzar una efectividad de la eliminación de los defectos de hasta un 90%.

Ahora, cuando las revisiones guiadas, las revisiones técnicas o las inspecciones son realizadas por un grupo de pares, la revisión puede llamarse una revisión por pares.

En nuestros proyectos, los consultores de RBCS/Business Innovations tienen una regla simple: Dos pares de ojos. Esto significa que, para cualquier producto del trabajo que sea importante, más de una persona lo examina antes de que se considere como terminado. Esto podría implicar una revisión informal para los informes de los defectos o una revisión formal para los casos de prueba, pero confiamos en el método de revisión para todo.

Un par de puntos tienen que ser adicionados aquí. Primero, cualquier producto de software o producto relacionado con el trabajo puede ser revisado. No sólo revise el código, los requisitos y las especificaciones del diseño. Revise todos los productos importantes del trabajo. Segundo, note que cualquier ítem el cual es revisado puede estar sujeto a más de una revisión. Usted puede realizar estas revisiones en cualquier orden que tenga sentido. Por ejemplo, usted puede realizar una revisión informal de una especificación del diseño antes de una revisión técnica de la especificación del diseño. Usted puede realizar una inspección de una especificación de requisitos

antes de una revisión guiada con los clientes.



Figura 3.1: Consenso y Entendimiento

Además de encontrar y eliminar los defectos, las revisiones pueden ayudar en el consenso y el entendimiento. La incompletitud y la ambigüedad pueden ocultar el verdadero significado de las especificaciones y una revisión puede revelar el significado. Podemos utilizar una revisión para alcanzar un acuerdo y entendimiento uniforme de las especificaciones.

Por ello, todos deben entender este objetivo. Una vez hicimos una evaluación donde la gente mencionó que si bien todos los requisitos del trabajo y las especificaciones del diseño pasaron por una revisión, algunas veces fueron realizadas después de que el código había sido escrito. Algunos participantes de la evaluación dudaron del valor de la revisión.

Cuando mencionamos esto a los interesados de la evaluación, ellos dijeron, "Bueno, la gente necesita entender que las revisiones ayudan a crear un acuerdo acerca de lo que se está construyendo".

Les contestamos: "Sí, eso tiene sentido, pero si los participantes no saben que éste es un objetivo de la revisión, ¿Pondrán ellos atención durante las revisiones?".

También es importante contar con personal de pruebas y aseguramiento de la calidad que atienda las revisiones cuando ellos puedan comprender el ítem sometido a revisión.

Revise esta cita del libro de Fred Brooks, *The Mythical Man Month*, el cual documenta su experiencia con las revisiones que se remontan a la década de los sesenta. "Mucho antes que cualquier código exista, la especificación debe ser entregada a un grupo de pruebas externo para que la completitud y claridad sean revisadas. Como dice V.A. Vyssotsky del proyecto Safeguard Project del Laboratorio Bell, los desarrolladores mismos no pueden realizar esto: "No te dirán que no lo entienden, inventarán felizmente su camino a través de las omisiones y oscuridades".

Los buenos probadores tienen la habilidad especial de reconocer las partes ambiguas, oscuras y faltantes de los requisitos y señalárlas en las revisiones.

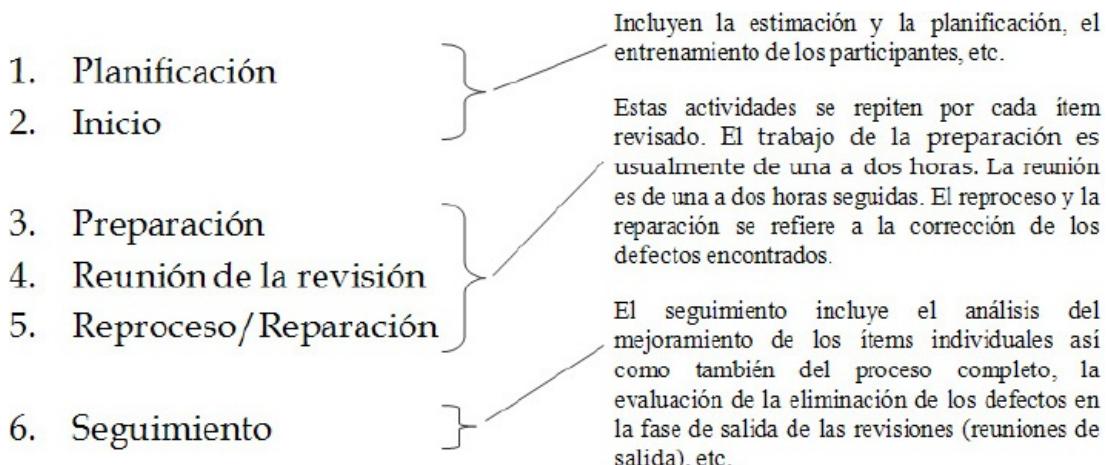


Figura 3.2: Un Proceso Genérico de Revisión

En esta figura, podemos observar un proceso genérico de la revisión. Consiste en seis pasos:

1. Planificación, que incluye la definición de los criterios de la revisión, la selección del personal, la asignación de los roles, la definición de los criterios de entrada y salida para los tipos de

- revisión más formales (p.ej. las inspecciones), la selección de las partes de los documentos que deben ser revisadas y la comprobación de los criterios de entrada (para los tipos de revisión más formales).
2. Inicio, que incluye la distribución de los documentos, la explicación a los participantes de los objetivos, el proceso y los documentos.
 3. Preparación individual, la cual incluye la preparación para la reunión de la revisión mediante la comprobación de los documentos y la observación de los defectos, las preguntas y los comentarios potenciales.
 4. La reunión misma de la revisión, la cual incluye las actividades relacionadas con las pruebas, la evaluación y la protocolización de los resultados. Estas actividades incluyen los resultados o los minutos de las argumentaciones y los registros (para los tipos de revisión más formales), la observación de los defectos, la creación de recomendaciones acerca del tratamiento de los defectos, la toma de decisiones en relación con los defectos, y las pruebas, la evaluación y las cuestiones de protocolización durante cualquier reunión física o el seguimiento de cualquiera de las comunicaciones electrónicas de grupo.
 5. Reproceso, el cual incluye la corrección de los defectos encontrados (realizados típicamente por el autor) y la protocolización de las actualizaciones del estado de los defectos (en revisiones formales).
 6. Seguimiento, que incluye la comprobación si los defectos han sido abordados, la recopilación de las métricas y la verificación de los criterios de salida (para los tipos de revisiones más formales).

La planificación, la definición de los criterios de entrada y salida y las actividades de inicio incluyen el trabajo para el proyecto entero y para cada ítem que tiene que ser revisado.

La comprobación de los criterios de salida, la preparación individual, la observación de los hallazgos, la reunión de revisión, el análisis de la reunión, el reproceso, la corrección de los defectos y el seguimiento de las actividades se repiten por cada ítem revisado. El trabajo de la preparación es usualmente de una a dos horas solamente. La reunión es de una a dos horas en total. El reproceso y la corrección de los defectos son realizados por el autor.

Sin embargo, el seguimiento no solo incluye el trabajo en los ítems individuales. El seguimiento incluye también el análisis del mejoramiento del proceso general, la evaluación de la eliminación de los defectos (o "bugs") en las revisiones de la salida de una fase (reuniones de salida), y etc.

Observe que los detalles del proceso de revisión dependen del tipo específico de revisión utilizado en el proyecto, así como también del tipo de revisión utilizado para cada clase particular del ítem.

Las revisiones implican que las personas desempeñen varios roles y asuman varias responsabilidades.

El moderador dirige las reuniones de revisión.

El escribano o secretario es la persona que reúne la información acerca de los hallazgos de la revisión.

El autor describe, explica y responde a preguntas acerca del ítem.

Los revisores o inspectores encuentran los defectos(o "bugs") en el ítem.

El jefe de proyecto planifica, organiza los recursos y la capacitación, apoya y analiza las métricas del proceso, pero, en algunos procesos, él no participa en la revisión.

Ahora, en algunos casos, una persona puede desempeñar múltiples roles. Por ejemplo, los autores actúan a veces como moderadores, como en la revisión guiada. Uno de los revisores puede actuar como secretario. Estos detalles son determinados por el tipo de revisión.

Si bien las revisiones son una forma muy eficiente de encontrar defectos, así como también una buena herramienta para la educación y la creación de consenso, no es trivial mantener una buena revisión. Para tener reuniones de revisión exitosas, podemos ofrecer las siguientes sugerencias:

Por un lado, proporcione capacitación para los participantes de la revisión. Esto no necesita ser extenso—una hora podría ser suficiente para las revisiones informales, aunque las técnicas formales como las inspecciones necesitarían mucho más—pero debería enseñar a la gente el proceso y las reglas básicas.

Asegúrese de revisar el producto, no el productor. Las reuniones de revisión no deberían convertirse en un foro para ataques personales de ningún tipo.

Como con cualquier reunión, usted debería establecer y seguir una agenda, y tener objetivos claros y formulados.

La mayoría de los métodos para las revisiones son acerca de la búsqueda de los problemas, en lugar de la identificación de las correcciones para ellas. En esos casos, debería limitar el debate acerca de los hallazgos de las personas. La excepción es que ciertos tipos de revisiones técnicas pueden incluir sesiones de lluvia de ideas para la solución de los problemas, las cuales deben tener períodos bien identificados dentro de la revisión.

Tenga un secretario o escribano cuyo trabajo sea de anotar, especialmente acerca de los problemas identificados. Usted debería tener algún tipo de proceso de ciclo cerrado para garantizar que todos

los problemas sean resueltos, incluso si es algo tan sencillo como pedir al autor que retorne una copia de la lista de los problemas con una marca de comprobación junto a cada uno cuando la corrección es realizada.

Ahora bien, podría recordar que en el capítulo 1, cuando hablamos de los beneficios de realizar el aseguramiento temprano de calidad y las pruebas tempranas, mencionamos que muy pocas organizaciones saben cuántos defectos son introducidos, detectados y eliminados en las primeras etapas del ciclo de vida. Si desea recopilar métricas acerca de las revisiones, tendrá que pensar en la forma adecuada de incorporar eso en este proceso. Algunas personas asumen que los mismos procesos, relativamente pesados, utilizados para hacer el seguimiento de los defectos durante los niveles de pruebas de etapas finales como las pruebas de sistema son los únicos métodos posibles. Usted puede utilizar esos métodos, pero a la mayoría de nuestros clientes no les agrada. Se recomienda identificar las métricas esenciales que desea recopilar de las revisiones y hacer el seguimiento sólo a éstas. Comprendiendo la manera de incluirlas en su base de datos del seguimiento de los defectos le permitirá extraer un conjunto unificado de las métricas de ese repositorio.

Debería limitar el número de personas allí y seleccionar cuidadosamente los participantes. Piense acerca de los objetivos de la revisión—recuerde, aquellos objetivos deberían ser claros y señalados — y pregúntese a usted mismo cómo cada participante contribuirá.

Todos deberían participar en la reunión de revisión preparada, lo que significa que han leído la revisión del ítem sometido a pruebas y han recopilado algunas notas preliminares. Usted puede asegurarse de la preparación, haciendo que la gente presente notas antes de la reunión.

Diferentes tipos de ítems tienen diferentes tipos de problemas y los diferentes ítems tendrán diferentes objetivos de la revisión. Por lo tanto, desarrolle una lista de comprobación para cada tipo de ítem que es revisado.

Revise las revisiones y sus resultados de forma periódica, para asegurarse de que el proceso está funcionando.

Utilice la técnica adecuada para cada tipo de ítem. Más antes mencionamos nuestra regla "dos pares de ojos" para los productos del trabajo de las pruebas. Esto significa que cada informe de defectos y cada caso de pruebas son revisados. Sin embargo, para los informes de los defectos, utilizamos una revisión rápida, muy informal, con sólo dos personas, mientras que para los casos de prueba utilizamos una revisión de pares más formal, con tres o cuatro personas.

Es muy útil de hacer participar a los probadores—si el probador puede leerlo—en las revisiones de los productos del trabajo así como los requisitos, los casos de uso, las especificaciones del diseño e incluso el código. Primero, el probador tiene una buena mentalidad para la revisión, especialmente encontrando los defectos. Segundo, los probadores pueden aprender más acerca del producto, durante el desarrollo temprano y utilizar ese conocimiento para crear los casos de prueba con más anticipación.

Evitar el uso indebido de los hallazgos y los resultados de las revisiones. Si la gente observa los resultados de la revisión utilizados para las evaluaciones del personal, la entrega de bonos, la determinación de aumentos de pagos o salarios, la asignación de culpabilidad y responsabilidad para los problemas del proyecto, o la entrega de cualquier recompensa o sanción relacionada con la gestión, la confianza será perdida y el proceso de revisión fallará—o se volverá tan desagradable o político que usted deseará que falle.

Como cualquier cambio del proceso lleva tiempo y requiere de recursos comprometidos, necesitará asegurarse del apoyo de la gerencia.

Por último, tenga en cuenta que las revisiones no son algo que aprende una vez y luego sabe perfectamente. El proceso de revisión debería ser mejorado continuamente.

Porque la mayoría de los probadores participarán en las revisiones de los requisitos y el diseño, aquí tiene algunos problemas comunes que ellos encontrarán.

Es bastante común encontrar ambigüedades, donde no es exactamente claro lo que significa una declaración o sección. Por ejemplo, considere una declaración como "El sistema debería permitir al usuario leer el correo electrónico de su Proveedor de Servicios de Internet". Está bien, de acuerdo, pero ¿Cuáles Proveedores de Servicios de Internet? ¿Todos? ¿Algunos de ellos? ¿Cuáles? ¿Qué tamaño de e-mails se permiten? ¿Cuáles tipos y tamaños de los archivos adjuntos se permiten?

También es común encontrar que los escenarios no han sido pensados cuidadosamente y tienen problemas de completitud que lo dejan a usted pensando, "Bueno, ¿Y qué pasa después?" Por ejemplo, considere una declaración como esta, "Al ingresar tres contraseñas inválidas, el sistema debería bloquear la cuenta del usuario". Esto parece una buena idea en un principio, desde el punto de vista de seguridad, pero pregúntese usted mismo algunas de las preguntas obvias. ¿Durante cuánto tiempo estará bloqueada la cuenta? ¿Cómo podemos desbloquearla antes si es necesario? ¿Quién está autorizado para desbloquearla? Esta declaración, aparentemente una mejora de la seguridad, en realidad podría permitir un pequeño e ingenioso ataque de la denegación del servicio. En primer lugar, el hacker ingresa tres contraseñas al azar para las cuentas administrativas o del súper usuario, bloqueando aquellas y luego procede a introducir las

contraseñas al azar para todas las otras cuentas. ¡Tantán! El software es inutilizable.

Un tercer problema común de los requisitos y el diseño es que la declaración no puede ser probada. Pregúntese: “¿Cómo puedo comprobar este requisito?” Si no hay manera de confirmar o negar el requisito, no es comprobable. Por ejemplo, considere la declaración, “El sistema debería proporcionar una disponibilidad del 100%”. Bueno, es posible realizar una prueba que cause una caída, para desaprobar esto, pero ¿Cómo podría confirmarlo? No existe una técnica de pruebas conocida para demostrar una disponibilidad perfecta del 100%. 99,999%, sí, pero no el 100%.

Por último, mantenga los ojos abiertos para las dependencias, el acoplamiento y la complejidad excesivos. Busque diagramas de diseño deformes y requisitos confusos. Si se le hace difícil de descubrirlos, es también probable que les sea difícil a los demás.

Ahora, examinemos el estándar que se aplica en las revisiones, el estándar IEEE 1028. Este estándar consiste en ocho secciones, de las cuales veremos las primeras cinco:

Sección 1, Descripción general, que abarca el propósito, el alcance, la conformidad, la organización y la aplicación del estándar.

Sección 2, Referencias

Sección 3, Definiciones

Sección 4, Revisiones de gestión, aborda las responsabilidades, las entradas y las salidas de los datos, los criterios de entrada y salida y los procedimientos para las revisiones de gestión. Tenga en cuenta que las revisiones de gestión no son parte del Programa de Estudios Nivel Básico.

Sección 5, Revisiones técnicas, también llamadas revisiones de pares, que cubren las responsabilidades, las entradas y salidas de datos, los criterios de entrada y salida y los procedimientos para estas revisiones.

Las tres secciones restantes del estándar IEEE 1028 son:

Sección 6, Inspecciones, las más formales de las revisiones, son cubiertas, con la información acerca de las responsabilidades, las entradas y las salidas de los datos, los criterios de entrada y salida y los procedimientos. Porque esta técnica es más formal que una revisión técnica, el estándar cubre también la recopilación de los datos y la mejora del proceso a través de las inspecciones.

Sección 7, Revisiones guiadas, que cubren las responsabilidades, las entradas y las salidas de los datos, los criterios de entrada y salida y los procedimientos. Las revisiones guiadas son menos formales que las inspecciones, pero, de acuerdo con el estándar IEEE 1028, más formales que las revisiones técnicas, así el estándar cubre también la colección de los datos y la mejora del proceso a través de las revisiones guiadas. En la práctica común, aunque una revisión guiada no es más que una revisión técnica, donde el autor es el moderador y el ítem sometido a revisión es la agenda.

Por último, la sección 8, las auditorías, abordan las responsabilidades, las entradas y las salidas de los datos, los criterios de entrada y salida y los procedimientos para este tipo de revisiones, los cuales están fuera del alcance del Programa de Estudios Nivel Básico.

Al igual que con el estándar IEEE 12207, el estándar CMMI y el estándar ISO 9126, los cuales fueron presentados anteriormente, se espera que recuerde este estándar en el examen. Le recomendamos que estudie este estándar cuidadosamente.

3.2.1 Ejercicios

Ejercicio 1

Forme equipos de 3 a 5 personas.²¹

Seleccione una técnica de revisión de las que ya hemos hablado. Si su técnica seleccionada involucra roles específicos, asigne los roles.

Revise el Documento de Requisitos de Marketing de Omninet.

Argumente acerca de los hallazgos de su equipo.

Solución del Ejercicio 1

Hemos resuelto este ejercicio en cursos en vivo en todo el mundo con cientos de personas. Típicamente, los equipos de revisión encuentran cerca de diez defectos de varios tipos en el Documento de los Requisitos de Marketing.

Encontramos más de veinte cuando realizamos este ejercicio por nosotros mismos (véase la tabla 3.1). Pudimos haber encontrado otros defectos—o defectos posibles—con este documento, pero la limitación del tiempo nos cortó. También encontré un número de defectos o defectos potenciales que necesiten probablemente ser resueltos en un documento de más bajo nivel, como el Documento de Requisitos del Sistema Omninet.

Una observación interesante, para lo que sirva, es que la mayoría de la gente en cada curso dijo

que el Documento de los Requisitos de Marketing de Omninet es tan bueno o mejor que los típicos documentos de requisitos que ellos reciben. Eso es verdad sin importar dónde en el mundo se ha presentado el curso. Si observa a través de los defectos que usted y nosotros encontramos en el Documento de los Requisitos de Marketing de Omninet, se puede imaginar cómo estos problemas podrían conducir después a problemas serios del sistema si es que no son corregidos antes de la implementación.

Sección #	Frase o Sección Problemática	Reformulación de la Frase o Sección
1.1	No todos los acrónimos fueron utilizados en el documento. Esto es confuso, porque no estamos seguros de por qué necesitamos saber acerca ellos.	Borrar los acrónimos no utilizados como son <i>AS, cable, CC, CS, DBMS, DC, IE, Linux, PIN, WS y WXP</i> .
1.2	Sería útil tener los prototipos de las pantallas.	Crear un documento separado de los prototipos de las pantallas.
3.1.2	¿Qué ocurre si el usuario declina la compra de más tiempo cuando se agota el tiempo?	Añadir la oración “Si el usuario declina la compra de más tiempo, el quiosco debería retornar al usuario a la sesión activa hasta que el período de tiempo actual ha expirado”.
3.1.2	¿Es diferente la terminación de la sesión basada en un reloj al cierre de la sesión (“logout”) en cuanto a la limpieza de los datos de la sesión? No debería serlo, pero los requisitos no lo dicen.	Añadir la oración, “El quiosco debería limpiar todos los datos de la sesión (como se describe en la sección 3.1.8) ante cualquier terminación de la sesión basada en la expiración del tiempo del reloj”.
3.1.2	¿Qué ocurre si el usuario se retira del quiosco durante el proceso inicial de pago? ¿Cuánto es un período razonable de tiempo de espera para retornar a la pantalla de Bienvenida?	Añadir el párrafo, “Si durante la secuencia inicial del pago el usuario no responde a ninguna alerta (p.ej. ‘Por favor cambie la tarjeta’, ‘Por favor inserte billetes’, ‘Por favor ingrese el período de tiempo que desea comprar’, etc.) dentro de los 30 segundos, el quiosco debería terminar la pantalla de pago y devolver cualquier dinero recibido durante esta secuencia de pago y retornar a la pantalla de Bienvenida”.
3.1.2	¿Qué ocurre si el tiempo del reloj expira mientras el usuario está comprando un período de tiempo adicional? ¿Cuánto es un período de tiempo de espera razonable para el proceso del pago del tiempo adicional? ¿Qué pasa cuando el usuario cambia de idea y se retira del quiosco?	Añadir la oración, “Si el presente período de tiempo expira mientras el usuario está comprando un período de tiempo adicional, el quiosco debería darle al usuario una notificación adicional y debería extender el tiempo del reloj por 15 segundos. Si el tiempo extendido del reloj expira antes de que el usuario haya completado la compra, el quiosco debería terminar la sesión del usuario”.
3.1.2	¿Tienen que volver a pasar sus tarjetas los clientes con tarjeta de crédito y débito para comprar tiempo adicional? Si no, considere los riesgos asociados con un cliente que se retira sin terminar su sesión.	Añada la oración, “Los clientes con tarjetas de crédito o tarjetas de débito deben volver a pasar sus tarjetas para comprar períodos de tiempo adicionales”.
3.1.2	No está claro si el límite de una hora acerca de las compras aplica a los períodos de tiempo adicionales comprados después de la ventana emergente de advertencia de 60 segundos.	Añada las oraciones, “El quiosco debería vender períodos de tiempo adicionales en incrementos de 5 minutos hasta una (1) hora por cada sesión. No se aplicarán limitaciones acerca de cuántas veces un usuario puede extender una sesión por medio de la compra de nuevos períodos de tiempo a raíz de la expiración del tiempo del reloj”.
3.1.3	Está dicho que la selección de los navegadores dependen del sistema operativo del quiosco, pero los sistemas operativos compatibles del	Añadir la oración, “Para prevenir que un gusano específico o epidemia de virus hagan caer toda la red de Omninet, el cincuenta por ciento de los quioscos deberían ser instalados con Linux,

	quiosco están implícitos, no especificados.	mientras que el otro cincuenta por ciento de los quioscos deberían ser instalados con Windows XP”
3.1.3	El quiosco tiene que ofrecer al usuario una selección de navegadores, ¿Pero no debería decir el requisito “navegadores alternativos”? Nosotros asumiríamos que todo el programa del quiosco sería una aplicación web ejecutándose en alguna especie de servidor Web local (p.ej., IIS o Apache), lo cual significaría que a excepción de los períodos de inicio o como un resultado de un error fatal, el software del quiosco se ejecutaría siempre en un navegador. Parecería que esta función proporciona el cambio a otros navegadores alternativos.	Cambie la frase para que se lea, “En la pantalla de Bienvenida, cada quiosco Omninet debería proporcionar al usuario una selección de navegadores alternativos. Estos deberían ser las últimas versiones de Netscape, Opera o Internet Explorer (disponibles sólo en los quioscos con Windows)”.
3.1.3	Si el comentario de arriba es correcto, entonces la falta de la especificación de navegadores alternativos versus por defecto para todos los sistemas operativos de los quioscos es una omisión en los requisitos, o en lo mínimo debería ser atendido en la especificación del diseño.	Añadir la frase, “Los quioscos con Windows deberían tener por defecto la última versión del navegador Internet Explorer. Los quioscos con Linux deberían tener por defecto la última versión del navegador Netscape”
3.1.5	El “idioma local principal” puede variar de un lado de la ciudad al otro. Por ejemplo, en San Antonio, Texas, donde vive Rex Black, cerca de la mitad de la población es hispana y el español es el idioma principal para mucha gente en ciertos barrios. Estudios demográficos de la ubicación real del quiosco—más que solo datos del censo para una ciudad o región—tendría que dirigir esta decisión para evitar quioscos inutilizados.	Cambiar la frase para que se lea, “Idioma local principal basado en la ubicación de la instalación y datos demográficos de los clientes potenciales que visitan el sitio real del quiosco”.
3.1.5	“En localidades donde comúnmente se hablan múltiples idiomas”, ¿Tendrá el usuario la opción de seleccionar su idioma preferido <i>en su idioma preferido</i> ?	Añadir una segunda oración a este párrafo, “El quiosco debería presentar cada idioma opcional al usuario <i>en ese idioma</i> ; (p.ej.: ‘Para español toque aquí’ en vez de ‘For Spanish, touch here’)”.
3.1.6	Las costumbres, las religiones locales y etc. influencian lo que es ofensivo, pero no hay reconocimiento de esta cuestión en esta sección.	Añada la oración “La obstaculización del contenido debería ser localizado para cada quiosco y debería tomar en cuenta las costumbres y las religiones locales, así como también las edades de los usuarios esperados del quiosco”.
3.2	Las referencias a los prototipos de las pantallas apropiados para la administración no se encuentran.	Añada las referencias apropiadas en cada subsección en 3.2.
3.2.1	Las actualizaciones automáticas deben ocurrir a las “2:00 AM en la hora local”, pero ésta es una hora que ocurre dos veces en un día en el año y cero en veces un día en el año en muchos lugares donde ocurren los cambios del horario por el ahorro de luz diurna o del reloj de verano.	Añadir la frase en el paréntesis, “(En el caso de un cambio de la hora local o basada en el cronograma [p.ej., los horarios de el ahorro de luz diurna] que podrían interferir con este cronograma, sin embargo el quiosco debería conectarse exactamente una vez por día lo más cerca que se pueda a las 2:00AM)”.
3.2.1	¿Cuál es el tiempo máximo permitido para una actualización?	Añadir la frase, “Los períodos de las actualizaciones no deberían exceder los 30 minutos. En el caso de que cualquier paquete de actualización individual provocara que la actualización exceda los 30 minutos, ese paquete debería ser dividido en una colección de paquetes de actualización priorizados

		más pequeños de manera que los quioscos puedan bajarlos e instalarlos dentro de este límite. Los quioscos deberían bajar e instalar aquellos paquetes de actualización en serie y en orden de prioridad”.
3.2.1	¿Está el quiosco disponible durante las actualizaciones? ¿Qué pasa si el quiosco está en uso cuando una actualización programada va a comenzar?	Añadir la frase, “El quiosco debería permanecer disponible para el uso durante las actualizaciones. El quiosco no debería terminar sesiones activas si es que una actualización se inicia durante la sesión. Durante la interacción activa del usuario con Internet, el proceso de actualización no debería consumir más de la mitad del ancho de banda del quiosco”.
3.2.3	¿Cuáles son las implicaciones de los riesgos de negocios acerca del acceso de los agentes de los centros de llamadas a la información sensible del cliente como los números de tarjetas de crédito, las direcciones de Internet visitadas, y así sucesivamente?	Añadir el párrafo, “Para minimizar las oportunidades de los agentes del centro de llamadas de usar indebidamente el acceso a la información sensible de los usuarios como los números de las tarjetas de crédito, las direcciones de Internet visitadas y así sucesivamente, las computadoras de escritorio y los servidores del centro de llamadas no deberían permitir las capturas de pantallas, el copiado y pegado, el guardado de archivos, los correos electrónicos o cualquier otra transferencia de datos desde el computador de escritorio del agente del centro de llamadas o los servidores del centro de llamadas a cualquier otro sistema, excepto cuando sea necesario satisfacer funciones especificadas”..

Tabla 3.1: Problemas con el Documento de los Requisitos de Marketing de Omninet

3.3 Análisis Estático por medio de Herramientas

Objetivos del Aprendizaje

LO-3.3.1 Recordar los defectos y los errores típicos identificados por el análisis estático y compararlos con las revisiones y las pruebas dinámicas. (K1)

LO-3.3.2 Describir, utilizando ejemplos, los beneficios típicos del análisis estático. (K2)

LO-3.3.3 Hacer una lista de los defectos típicos del código y el diseño que pueden ser identificados por las herramientas de análisis estático. (K1)

Esta sección, Análisis estático por medio de herramientas, cubrirá los siguientes conceptos clave:

- Defectos típicos y errores identificados por el análisis estático en comparación con las revisiones y las pruebas dinámicas.
- Beneficios típicos del análisis estático.
- Los típicos defectos de código y diseño identificados por las herramientas de análisis estático.

Empecemos a comparar y contrastar el análisis estático con las pruebas dinámicas.

Al igual que las pruebas dinámicas, el análisis estático busca defectos en el software mismo, pero también puede buscar defectos en los modelos del software como los requisitos o los modelos ejecutables así como los modelos de rendimiento del sistema que mencionamos anteriormente.

En un contraste adicional, a diferencia de las pruebas dinámicas, el análisis estático se realiza sin realmente ejecutar el sistema, más bien, se ejecuta una herramienta que verifica el sistema o un modelo de éste.

Análisis estático implica el análisis del sistema o sus componentes por una herramienta—eso es lo que hace diferente al análisis estático de una revisión, la cual es manual. Las pruebas dinámicas no siempre involucran herramientas

El análisis estático puede encontrar defectos que son difíciles de encontrar o aislar en las pruebas dinámicas.

Los ejemplos incluyen cuestiones de mantenibilidad con el código, que podría ejecutarse bien, pero podría ser difícil de entender (y modificar), así como también la utilización insegura de punteros, que podrían causar una caída o un comportamiento extraño sólo en circunstancias especiales.

Por último, tenga presente que el aislamiento del defecto es más fácil porque usted encuentra el defecto, no el síntoma o la falla.

¿Qué puede ser objeto del análisis estático? Bueno, muchas cosas pueden, por lo general más de lo esperado.

Generalmente, las personas piensan sólo en el código del programa, examinando los flujos de

control y el flujo de datos, examinando las violaciones de los estándares de codificación. Sí, ése es un uso común del análisis estático, pero no el único.

Glosario del ISTQB

Flujo de control: Una secuencia de eventos (caminos) en la ejecución a través de un componente o sistema.

Flujo de datos: Una representación abstracta de la secuencia y los cambios posibles del estado de los objetos de datos, donde el estado de un objeto puede ser cualquiera de los siguientes: creación, uso o destrucción.

Hay simuladores que nos permiten analizar los modelos del programa, así como los modelos de rendimiento.

Podemos comprobar las salidas generadas como HTML y XML acerca de la conformidad con una sintaxis correcta, los enlaces rotos, y así sucesivamente.

Por último, incluso los análisis usuales, como la ortografía, la gramática, las comprobaciones de los requisitos acerca de la dificultad de su lectura y la legibilidad y la claridad de los documentos del diseño.

Las herramientas de análisis estático no son necesariamente económicas, e incluso cuando lo son, las personas no siempre se toman el tiempo para usarlas, porque no entienden los beneficios del análisis estático.

Al igual que con las revisiones, el análisis estático proporciona una detección temprana y más económica de los defectos, a menudo mucho antes que la ejecución de las pruebas comience.

Esto significa que los defectos pueden ser encontrados y eliminados en muchos casos **fueras** de la ruta crítica para la versión, a diferencia de los defectos encontrados y eliminados durante los niveles de pruebas en las últimas etapas, como ser las pruebas de sistema y las pruebas de aceptación.

Los análisis estáticos pueden proporcionar advertencias acerca de dónde podrían existir agrupaciones de defectos, debido a la programación peligrosa, la alta complejidad y así sucesivamente.

Los análisis estáticos pueden localizar defectos que las pruebas dinámicas no podrían encontrar, así como el código difícil de mantener, complejo o ilegible.

El análisis estático puede detectar dependencias e inconsistencias en los modelos del software. Por ejemplo, enlaces rotos en las páginas Web.

Estos beneficios se combinan para ayudarnos a mejorar la mantenibilidad del código y los diseños, y, si recolectamos las métricas y estudiamos las lecciones aprendidas del análisis estático, nos puede ayudar a prevenir defectos.

Los problemas típicos encontrados durante el análisis estático incluyen:

Referencia a una variable con un valor no definido, que puede hacer caer al sistema si la variable es un puntero o un resultado erróneo, si es que son utilizados valores aleatorios en un cálculo.

Interfaces inconsistentes entre los módulos y componentes, así como el paso de un entero donde se requiere un número real.

VARIABLES que nunca son utilizadas, así perdiendo espacio de memoria y código inalcanzable (o muerto), lo cual no sólo desperdicia espacio, sino podría crear riesgos de seguridad.

Lógica faltante o incorrecta, así como la utilización del operador de asignación en lugar del operador de igualdad lógica en la condición de una sentencia if.

Complejidad excesiva del código, tal vez utilizando una métrica como por ejemplo la complejidad ciclomática, acerca de la cual hablaremos más adelante en este libro.

Las violaciones de los estándares de programación son otro defecto común del análisis estático, y muchas herramientas comerciales de análisis estático incluyen bibliotecas enteras de estándares de codificación.

Algunas herramientas de análisis estático ayudarán también a encontrar especialmente vulnerabilidades de seguridad, las cuales son tipos particulares de violaciones de los estándares de codificación.

Por último, el análisis estático puede localizar las violaciones de sintaxis de código y de los modelos del software.

¿Cómo y quién utiliza las herramientas de análisis estático?

Si estamos hablando del código, entonces los usuarios típicos son los programadores, con frecuencia durante las pruebas de componente e integración, aunque ellos comenzarían idealmente durante la codificación.

Si hablamos de diseño, entonces los usuarios típicos son los diseñadores y los arquitectos durante el período del diseño. El modelado del rendimiento que hemos mencionado unas cuantas veces suele ocurrir en ese punto.

Ahora, una cosa que hemos encontrado con nuestros clientes es que durante la presentación inicial de las herramientas de análisis del código en una base de código existente, estas herramientas

pueden producir un gran número de mensajes de advertencia. Por lo tanto, le recomendamos el siguiente proceso:

- Determine cuáles reglas hará valer y cuáles no son importantes. Desactive las que no son importantes.
- Exija la utilización de las herramientas de análisis estático, pero sólo en funciones o clases nuevas o aquellas funciones o clases que están siendo modificadas.
- Haga que el programador ejecute el análisis estático y **sus pruebas de unidad** antes de declarar el código como terminado y exíjale que presente los resultados del análisis estático y las pruebas de unidad, junto con las pruebas unitarias mismas, para una revisión del código antes de aceptar el código como terminado.

Si este proceso se aplica con diligencia en el tiempo, las partes más defectuosas del código existente—siendo las más propensas que necesitan mantenimiento—se pondrán gradualmente en conformidad con los estándares de codificación. En cuanto al resto del código, oiga, si no está causando problemas, creo que las violaciones de los estándares de codificación que existen no están creando demasiados problemas.

Algunas personas utilizan los compiladores para hacer su análisis estático, pero hay muchas herramientas sofisticadas. Sin embargo, algunos compiladores y entornos de desarrollo integrados son también cada vez más sofisticados en este sentido.

Glosario del ISTQB

Compilador: Este término no está definido en el glosario del ISTQB. La política del ISTQB es que los términos no definidos en el glosario no pueden ser incluidos en el examen.

3.3.1 Ejercicios

Ejercicio 1

¿Cuál tipo de análisis estático sugeriría para Omninet?

¿Utilizaría el análisis estático en áreas que estarían sujetas a pruebas posteriores?

Argumente.

Solución del Ejercicio 1

En Omninet, utilizaríamos el análisis estático para el código Java, C++ u otro lenguaje de programación utilizado para crear las aplicaciones en los servidores y el procesamiento de los pagos y los períodos de tiempos de los quioscos. También utilizaríamos el análisis estático en HTML en las interfaces del usuario, los repositorios de datos XML y las bases de datos SQL.

Además, utilizaríamos modelos temprano en el proyecto para predecir el rendimiento y la reacción a la carga. Utilizaríamos ambos, los modelos estáticos como una hoja de cálculo y los modelos dinámicos como un simulador del rendimiento del sistema de redes. Para Omninet, los problemas del rendimiento que indicaron cuestiones de arquitectura sería desastroso si aparecen durante la prueba de sistema. Por cierto, trabajamos en un proyecto que falló debido al descubrimiento tardío de cuestiones de rendimiento básico. También trabajamos en un proyecto donde utilizamos modelos estáticos y dinámicos para evitar cualquier descubrimiento tardío de esas cuestiones de rendimiento.

Incluso utilizaríamos comprobadores de ortografía y gramática en las especificaciones de los prototipos de la interfaz del usuario, los requisitos y el diseño y los planes de pruebas y los proyectos. Usted podría preguntar “¿Cuáles problemas importantes podría encontrar un comprobador de gramática?” Si el plan de pruebas incluye demasiadas oraciones con verbos pasivos como, “Tantos defectos como sea posible deberían ser detectados”, sin especificar quién debe hacer qué para detectar esos defectos, es un plan deficiente, uno que no puede ser puesto en acción.

Mientras que somos grandes partidarios del análisis estático, también probaríamos cada uno de los ítems que pasaron por el análisis estático. Asuma que el análisis estático es 65% efectivo en promedio y que ha encontrado y eliminado 650 defectos a través del análisis estático de cada parte de Omninet. Esto significa que 350 defectos han escapado y deben ser detectados y eliminados durante las pruebas.²²

Preguntas de Examen de Muestra y Simulación

Para finalizar cada capítulo, usted puede tratar de resolver una o más preguntas de examen de muestra para reforzar su conocimiento y comprensión del material y prepararse para el examen del Probador ISTQB Nivel Básico.

Sección 3.1 Técnicas estáticas y el proceso de pruebas (K2)

Términos

Pruebas dinámicas y pruebas estáticas.

#	Pregunta	K
66.	<p>Objetivo del aprendizaje: LO-3.1.1 Considere los siguientes productos del trabajo del software:</p> <ul style="list-style-type: none"> I. Plan de prueba. II. Plan de proyecto. III. Especificación del diseño de pruebas IV. Especificación del diseño del sistema. V. Código fuente. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Todos estos productos del trabajo pueden ser revisados. B. De todos los productos del trabajo en un proyecto de software, sólo estos pueden ser revisados. C. Sólo II, IV, y V pueden ser revisados. D. Sólo I y III pueden ser revisados. 	1
67.	<p>Objetivo del aprendizaje: LO-3.1.2 Usted está trabajando como un probador en un proyecto grande. Usted es invitado a asistir en una reunión acerca de la revisión de la especificación de los requisitos como el representante del equipo de pruebas. ¿Qué razón personal podría convencerlo a usted para asistir en esta reunión?, o, cambiando la pregunta de otra forma, ¿Cuáles beneficios de las revisiones afectan a las pruebas más directa e inmediatamente?</p> <ul style="list-style-type: none"> A. El número inferior de defectos entregados a los clientes después de la versión. B. El número inferior de defectos entregados a los probadores durante la ejecución de las pruebas. C. La conclusión más rápida del proyecto. D. El costo reducido del soporte del producto. 	2
68.	<p>Objetivo del aprendizaje: LO-3.1.3 Usted ha instalado el software sometido a pruebas en su computadora. Está ejecutando el software y pasando por las pantallas para revisar que cada indicador, incluyendo los mensajes de error estén correctos. ¿Qué técnica de prueba está utilizando?</p> <ul style="list-style-type: none"> A. Exploratoria. B. Relacionada el cambio. C. Estática. D. Dinámica. 	2
69.	<p>Objetivo del aprendizaje: término. ¿Qué es una revisión?</p> <ul style="list-style-type: none"> A. Una evaluación del estado de un producto o proyecto para determinar las discrepancias de los resultados planificados y para recomendar mejoras. B. Las pruebas de un componente o sistema en el nivel de especificación o implementación sin la ejecución de ese software. C. Las pruebas que utilizan un modelo de las operaciones del sistema y su probabilidad de utilización típica. D. Una presentación paso a paso por el autor de un documento. 	1

Sección 3.2 Proceso de revisión (K2)

Estándares

- [IEEE 1028] Estándar IEEE 1028™ (1997) Estándar IEEE para las Revisiones de Software].

Términos

Criterios de entrada, revisión formal, revisión informal, inspección, métrica, moderador/líder de la inspección, revisión por pares, revisor, revisión técnica y revisión guiada.

#	Pregunta	K
70.	<p>Objetivo del aprendizaje: LO-3.2.1 En una revisión, ¿Cuál es el nombre de la función desempeñada por personas con antecedentes técnicos o de negocios específicos quienes identifican y describen los</p>	1

	<p>defectos en el producto bajo revisión?</p> <ol style="list-style-type: none"> Jefe. Revisor. Moderador. Autor. 	
71.	<p>Objetivo del aprendizaje: LO-3.2.2</p> <p>¿Cuál de las siguientes es la diferencia principal entre una revisión guiada y una inspección?</p> <ol style="list-style-type: none"> En una inspección, el autor da una presentación paso a paso del documento y dirige la reunión de la revisión. No hay diferencia; los términos son sinónimos. En una revisión guiada, el autor da una presentación paso a paso del documento y dirige la reunión de la revisión. Una revisión guiada es una forma de programación por pares, mientras que una inspección puede ser aplicada a cualquier producto del trabajo. 	2
72.	<p>Objetivo del aprendizaje: LO-3.2.3</p> <p>Para tener revisiones exitosas, es crítico que la gente esté motivada para encontrar defectos en el producto del trabajo bajo revisión.</p> <p>¿Por qué?</p> <ol style="list-style-type: none"> Experiencias negativas en las revisiones motivarán a los autores a cometer menos equivocaciones en el futuro. Las métricas de las revisiones deberían ser utilizadas por los jefes para tasar el rendimiento de los empleados. Porque que alguna serie de defectos están ciertamente presentes, es mejor eliminarlos más temprano y más económico en una revisión. El hallazgo de los defectos en el trabajo de cada uno ayuda a crear un ambiente de trabajo humilde y sin ego. 	2
73.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es una revisión por pares?</p> <ol style="list-style-type: none"> Una revisión de un producto del trabajo de software por los colegas del autor del producto con el propósito de identificar defectos y mejoras. Un examen formal, visual de los documentos para detectar defectos, basado en un procedimiento documentado. Una presentación paso a paso por el autor de un documento para recolectar información y establecer un entendimiento común de su contenido. Una evaluación sistemática de la adquisición, suministro, desarrollo, operación o del proceso de mantenimiento del software 	1
74.	<p>Objetivo del aprendizaje: Estándar IEEE 1028.</p> <p>¿Cuál estándar que es referenciado en el Programa de Estudios del ISQTB Nivel Básico de 2007, se encarga de las revisiones de gestión, las revisiones técnicas, las revisiones guiadas, las inspecciones y las auditorías?</p> <ol style="list-style-type: none"> IEEE 829. CMMI. ISO 9126. IEEE 1028. 	1

Sección 3.3: Análisis estáticos por herramientas (K2)

Términos

Compilador, complejidad, flujo de control, flujo de datos y análisis estático.

#	Pregunta	K
75.	<p>Objetivo del aprendizaje: LO-3.3.1</p> <p>¿Cuál es la diferencia clave entre el análisis estático y las pruebas dinámicas?</p> <ol style="list-style-type: none"> El análisis estático no implica la ejecución del código, mientras que las pruebas dinámicas si lo hacen. Las pruebas dinámicas no implican la ejecución del código, mientras que el análisis estático si lo hace. No hay diferencia; los términos son sinónimos. Las pruebas dinámicas encuentran defectos, mientras que el análisis estático encuentra fallas. 	1

76.	<p>Objetivo del aprendizaje: LO-3.3.1</p> <p>¿Cuál de los siguientes es un típico defecto identificado por el análisis estático?</p> <ul style="list-style-type: none"> A. El lanzamiento de una excepción no controlada. B. La caída de la aplicación. C. La referencia de una variable con un valor indefinido. D. La baja complejidad del código fuente. 	1
77.	<p>Objetivo del aprendizaje: LO-3.3.2 1 Considere lo siguiente:</p> <ul style="list-style-type: none"> A. La detección temprana de los defectos. B. Las métricas de productividad acerca de los programadores. C. Las advertencias tempranas acerca del código de las métricas. D. La prevención de los defectos. E. La detección de los requisitos faltantes. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Todos son beneficios típicos del análisis estático del código. B. I, III, y IV son beneficios típicos del análisis estático del código. C. I, II, y V son beneficios típicos del análisis estático del código. D. II y V son beneficios típicos del análisis estático del código. 	1
78.	<p>Objetivo del aprendizaje: LO-3.3.3</p> <p>¿Cuál de los siguientes es un típico defecto del código que una herramienta de análisis estático podría encontrar?</p> <ul style="list-style-type: none"> A. Una vulnerabilidad de seguridad relacionada con un desbordamiento de búfer. B. El problema de la interfaz de usuario con el esquema de colores. C. El código faltante para un caso de uso clave. D. La posibilidad de empezar una tarea la cual implicaría un esfuerzo inútil. 	1
79.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es complejidad?</p> <ul style="list-style-type: none"> A. El número de los caminos independientes a través de un programa. B. El porcentaje de los defectos detectados antes de la versión. C. El costo total del proyecto dividido entre el número de las líneas de código nuevas o modificadas. D. El grado en el cual un componente o sistema tiene un diseño y/o estructura interna que es difícil de comprender, mantener y verificar. 	1

Capítulo 3 Pregunta a través de las secciones

#	Pregunta	K
80.	<p>Cubre 3.2 y 3.3</p> <p>¿Cuál de las siguientes sería una diferencia clave entre una revisión del código por pares y un análisis estático del código utilizando una herramienta?</p> <ul style="list-style-type: none"> A. El análisis estático puede ser realizado sólo por el autor. B. Una revisión por pares encuentra defectos mientras que el análisis estático encuentra fallas. C. Una revisión por pares encuentra fallas mientras que el análisis estático encuentra defectos. D. Las revisiones por pares no pueden encontrar los requisitos faltantes mientras que el análisis estático puede. 	2

Preguntas del Examen de Simulación 1

#	Pregunta	K
81.	<p>Objetivo del aprendizaje: LO-3.1.2</p> <p>Usted está trabajando en un proyecto en el cual cualquier desfase en la fecha de la finalización de la ejecución de las pruebas resultaría en penalidades financieras para su empleador. El logro de la fecha de entrega planificada es visto como la única más alta prioridad. ¿Cuál beneficio de las técnicas estáticas es el que más probablemente convenza a la gerencia de utilizarlas?</p> <ul style="list-style-type: none"> A. Los costos reducidos del soporte después de la versión. B. La comunicación mejorada dentro del equipo. 	2

	C. Las mejoras de la productividad. D. La reducción del costo y tiempo de las pruebas.	
82.	Objetivo del aprendizaje: término ¿Qué es el análisis estático? A. Una actividad de debate de grupo por pares que se enfoca en lograr consenso acerca del método técnico que debe ser tomado. B. El análisis de los artefactos del software, p.ej. los requisitos del código, llevados a cabo sin la ejecución de estos artefactos del software. C. Una presentación paso a paso de un documento por el autor para recolectar información y establecer un entendimiento común de su contenido. D. Una revisión caracterizada por los procedimientos y requisitos documentados.	1
83.	Objetivo del aprendizaje: LO-3.2.1 1 ¿Cuál de los siguientes roles no es un rol desempeñado en una revisión formal típica? A. Revisor. B. Moderador. C. Analista de negocios. D. Autor.	1

Preguntas del Examen de Simulación 2

#	Pregunta	K
84.	Objetivo del aprendizaje: Estándar IEEE 1028. Considere la siguiente lista: I. Revisiones de gestión. II. Especificaciones de salida. III. Refinamientos del método. IV. Inspecciones. V. Auditorías. ¿Cuál de las siguientes afirmaciones es verdadera? A. I, IV, y V son secciones importantes en el estándar IEEE 1028. B. Todas son secciones importantes en el estándar IEEE 1028. C. I y V son secciones importantes en el estándar IEEE 1028. D. II y III son secciones importantes en el estándar IEEE 1028.	1
85.	Objetivo del aprendizaje: LO-3.3.1 Usted está utilizando herramientas de análisis estático para comprobar el código fuente de los varios tipos de errores de programación. ¿Cuál podría ser su objetivo? A. Encontrar las fallas en la funcionalidad importante. B. Encontrar los defectos en el código fuente. C. Encontrar los requisitos faltantes. D. Demostrar la aptitud para el uso.	1
86.	Objetivo del aprendizaje: LO-3.1.3 1 ¿Cuál de las siguientes afirmaciones es verdadera acerca de las pruebas estáticas? A. Las pruebas estáticas implican la ejecución del software sometido a pruebas. B. Las pruebas estáticas son siempre realizadas utilizando una herramienta. C. Las pruebas estáticas no encuentran los defectos a un costo más económico que las pruebas dinámicas. D. Las pruebas estáticas no implican la ejecución del software sometido a pruebas.	1

²⁰ Véase el libro de Capers Jones, *Estimating Software Costs*, para las cifras acerca de la efectividad típica de varias actividades del aseguramiento de calidad y las pruebas.

²¹ Este ejercicio y su solución han sido adaptados del capítulo 9 del libro de Rex Black *Pragmatic Software Testing*

²² El libro *Estimating Software Costs* de Capers Jones cita el 65% como la efectividad de las

revisiones de diseño y código, entonces estamos utilizando ese cálculo como una estimación aproximada de la efectividad del análisis estático solo para ilustrar este punto.

Capítulo 4

Técnicas de Diseño de Pruebas

"Por norma, los sistemas software no funcionan bien hasta que han sido utilizados y han fallado repetidamente en entornos reales"

Dave Parnas, pionero de la ingeniería del software.

El capítulo 4, Técnicas de Diseño de Pruebas, contiene las siguientes 6 secciones:

1. Proceso de desarrollo de pruebas.
2. Categorías de las técnicas de diseño de pruebas.
3. Técnicas basadas en la especificación o de caja negra.
4. Técnicas basadas en la estructura o de caja blanca.
5. Técnicas basadas en la experiencia.
6. Selección de las técnicas de pruebas.

Cada una de las secciones estará desglosada en dos o más partes.

4.1 Proceso de Desarrollo de Pruebas

Objetivos del Aprendizaje

- LO-4.1.1 Diferenciar entre una especificación del diseño de pruebas, especificación de casos de prueba y especificación de procedimiento de pruebas. (K2)
- LO-4.1.2 Comparar los términos condición de prueba, caso de prueba y procedimiento de prueba. (K2)
- LO-4.1.3 Evaluar la calidad de los casos de prueba en términos de trazabilidad clara a los requisitos y resultados esperados. (K2)
- LO-4.1.4 Traducir los casos de prueba en una especificación de procedimiento de prueba bien estructurada en un nivel de detalle relevante para el conocimiento de los probadores. (K3)

Glosario del ISTQB

Especificación de casos de prueba: Un documento que especifica un conjunto de casos de prueba (objetivo, entradas, acciones de pruebas, resultados esperados, y las precondiciones de ejecución) para un ítem de prueba.

Diseño de pruebas: (1) Véase la especificación de diseño de pruebas. (2) El proceso de transformación de los objetivos generales de pruebas en condiciones de pruebas tangibles y casos de prueba.

Especificación del diseño de pruebas: Un documento que especifica las condiciones de prueba (ítems de cobertura) para un ítem de prueba, el método de pruebas detallado y la identificación de los casos de prueba asociados de alto nivel. Tenga en cuenta que este término no se enunció específicamente para esta sección, pero se lo incluye aquí, ya que es un sinónimo de diseño de pruebas.

Especificación del procedimiento de prueba: Un documento que especifica una secuencia de acciones para la ejecución de una prueba. También conocido como guión de prueba o guión de prueba manual.

Trazabilidad: La habilidad de identificar ítems relacionados en la documentación y el software, tales como los requisitos con las pruebas asociadas. Véase también la trazabilidad horizontal, trazabilidad vertical.

Esta sección, proceso de desarrollo de pruebas, cubrirá los siguientes conceptos clave:

- Las pruebas deben basarse en el análisis de los riesgos.
- El nivel del riesgo está determinado por la probabilidad y el impacto.
- Podemos seguir un proceso sistemático para especificar los diseños, los casos y los procedimientos de prueba.
- Los casos de prueba, los datos existentes y los resultados esperados, son elaborados o traducidos en procedimientos de prueba.
- Así, los casos de prueba están relacionados con o son trazables a los procedimientos específicos de prueba.
- Podemos utilizar el riesgo para desarrollar el cronograma de la ejecución de pruebas.

Glosario del ISTQB

Cronograma de ejecución de pruebas: Un esquema para la ejecución de los procedimientos de prueba. Los procedimientos de prueba están incluidos en el cronograma de ejecución de pruebas en su contexto y en el orden en el cuál tienen que ser ejecutados.



Figura 4.1: Fases del Desarrollo de Pruebas

El desarrollo o la especificación de las pruebas que deben ser ejecutadas pueden llevarse a cabo en una secuencia sistemática de fases. El diagrama en esta figura muestra esos métodos, asumiendo que usted está utilizando una estrategia analítica de pruebas basada en los riesgos.

En primer lugar, tenemos el análisis para identificar las condiciones que deben ser probadas. Si está utilizando una estrategia analítica de pruebas basada en los riesgos, debería utilizar un análisis de los riesgos de calidad. Tenga en cuenta que los planes de proyecto, los requisitos de producto y la especificación de diseño del sistema son todos datos de entrada en el proceso del análisis de los riesgos de calidad. Nosotros mantenemos la trazabilidad hacia atrás a los documentos fuentes—es decir, al plan de proyecto, los requisitos o el diseño— para los riesgos de calidad que surgen de las declaraciones específicas en aquellos documentos.

A continuación, se produce el diseño de pruebas de nivel alto. Además refinaremos las condiciones de pruebas, creando una especificación del diseño de pruebas. Utilizamos las especificaciones del diseño del sistema y las especificaciones detalladas del diseño como un dato de entrada en este proceso. También nos referimos a las pruebas existentes, si las hubiere, para buscar oportunidades de reutilización y también para adherirlas a los estándares existentes, las granularidades y los estilos del diseño de pruebas.

Producimos un juego de pruebas, un framework lógico para los casos de prueba. Además capturamos la trazabilidad hacia atrás a la base de pruebas—en este caso los riesgos de calidad.

Por último, el diseño de pruebas de nivel bajo o la implementación de las pruebas son llevadas a cabo. Refinamos o elaboramos el diseño de pruebas en casos de prueba y luego refinamos aquellos casos de prueba en procedimientos de prueba, también llamados guiones de prueba. Otra vez, nos referimos a las especificaciones detalladas del diseño como una entrada. Utilizamos también las pruebas existentes para la reutilización y los estándares, como anteriormente. Las versiones tempranas del objeto de pruebas nos permiten probar nuestras pruebas, si es que están disponibles. Y nuevamente, mantenemos la trazabilidad de los casos y procedimientos de prueba hacia atrás a los riesgos de calidad.

Tenga en cuenta que, en la medida de lo posible, hemos utilizado entradas externas como los planes del proyecto y los requisitos para ayudarnos a crear los entregables internos, el testware. Una estrategia de pruebas basada netamente en los requisitos necesita estos entregables. Una estrategia analítica basada en los riesgos puede ser llevada a cabo sin dicha documentación, sólo con la participación de los interesados del negocio en un análisis de los riesgos. Sin embargo, la utilización de los requisitos y los documentos del diseño conducen a análisis de riesgos más precisos.

Examinemos este proceso en acción.

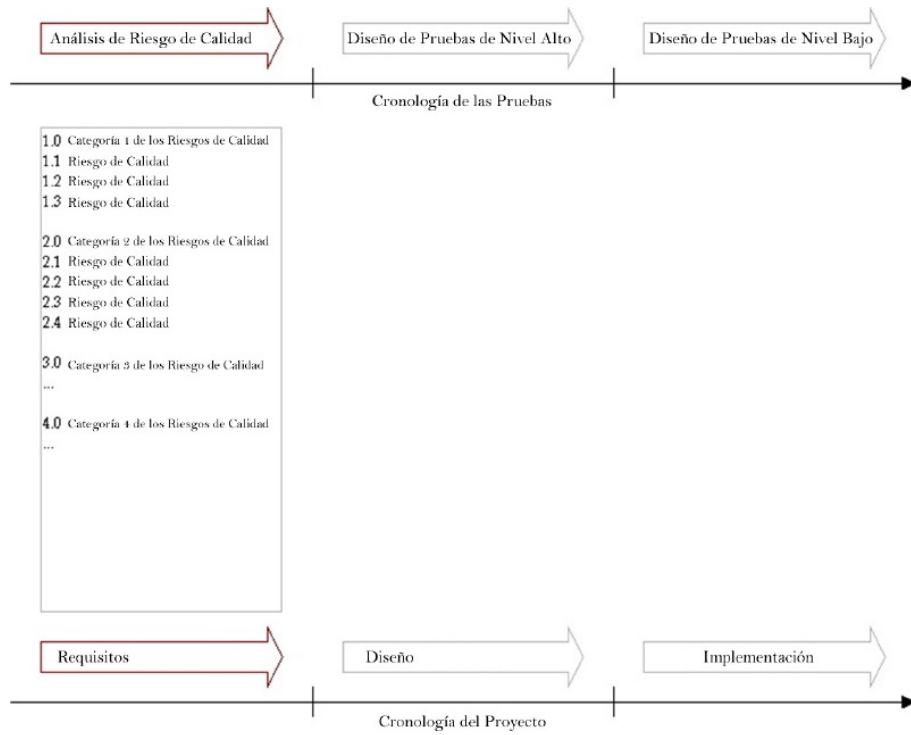


Figura 4.2: Análisis de Riesgos de Calidad

Nosotros comenzamos con el análisis de las pruebas.

Idealmente, al utilizar una estrategia analítica de pruebas para desarrollar las pruebas, el proceso del análisis ocurrirá en paralelo con la especificación de los requisitos. Esto es mostrado en esta figura por la cronología de las pruebas paralelas y el proyecto en la parte superior e inferior de la figura, respectivamente.

Aquí vemos los riesgos de calidad identificados y colocados en categorías. Vamos a explicar más este proceso del análisis de los riesgos de calidad en la siguiente parte de esta sección.

Siguiendo una estrategia basada en los requisitos, es similar en cuanto a la coordinación de los tiempos. Sin embargo, en ese caso, usted identifica las condiciones de las pruebas únicamente a partir de la especificación de los requisitos.

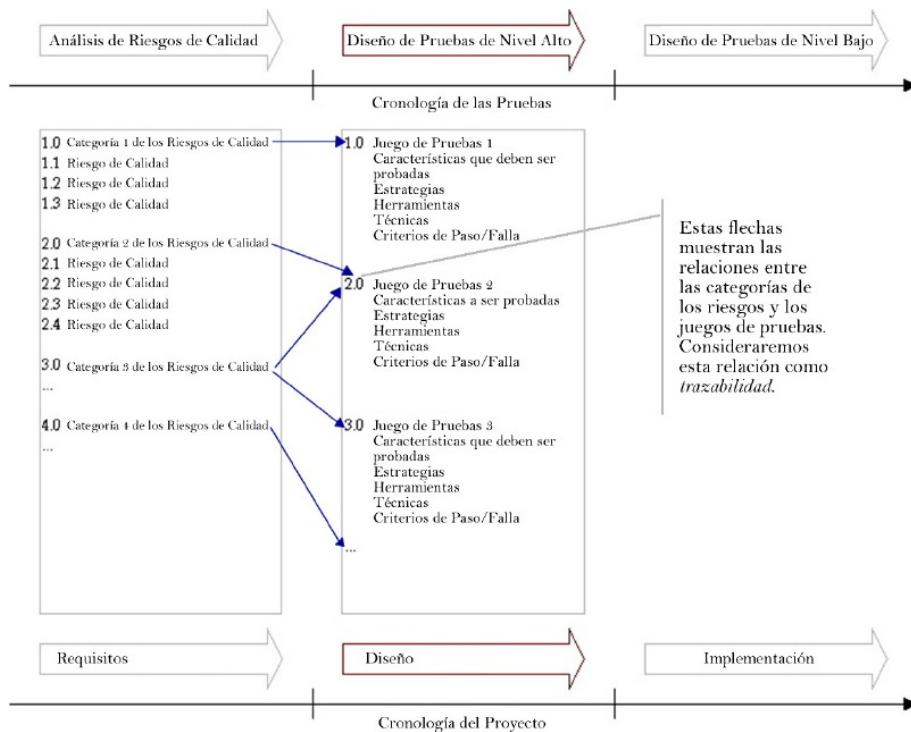


Figura 4.3: Diseño de Pruebas de Nivel Alto

Pasemos al diseño de las pruebas de alto nivel.

Una vez más, en condiciones ideales, el diseño de las pruebas de alto nivel se produce en paralelo

con el diseño del sistema. Eso se muestra nuevamente en esta figura como referencia.

Esta figura muestra que hemos identificado juegos de pruebas, basado en nuestro análisis de riesgos de calidad. Un juego de pruebas es una colección lógica de casos de prueba. Por ejemplo, usted puede tener un juego de pruebas de rendimiento, un juego de pruebas funcionales, un juego de pruebas de manejo de errores, y así sucesivamente.

Para cada juego de pruebas, identificamos las condiciones de las pruebas y las características que deben ser probadas. Esto es la parte del “qué” del juego de pruebas. Para cada juego de pruebas, identificamos también el “cómo”, las estrategias, las herramientas y las técnicas que utilizaremos para probar estas características.

Además, especificamos los criterios de paso/falla, los cuales son los medios por los que vamos a determinar si una prueba pasó o falló. Por ejemplo, en el caso de un juego de pruebas de rendimiento, podríamos tener una referencia a una declaración de requisitos, que dice, que todas las actualizaciones de la pantalla deben ocurrir en medio segundo o menos, cuando un usuario ingresa una entrada. Estos criterios de paso/falla son también conocidos como “oráculos de pruebas”.

Observe en esta figura cómo capturamos la información de la trazabilidad de los juegos de pruebas hacia atrás a las categorías de riesgos. Un juego de pruebas podría relacionarse con múltiples categorías de riesgos. Un juego de pruebas de rendimiento podría relacionarse hacia atrás con las categorías de riesgos de carga y tiempo de respuesta. Una categoría de riesgos también podría relacionarse con múltiples juegos de pruebas. Una categoría de riesgos de carga podría relacionarse con el juego de pruebas de rendimiento y el juego de pruebas de robustez.

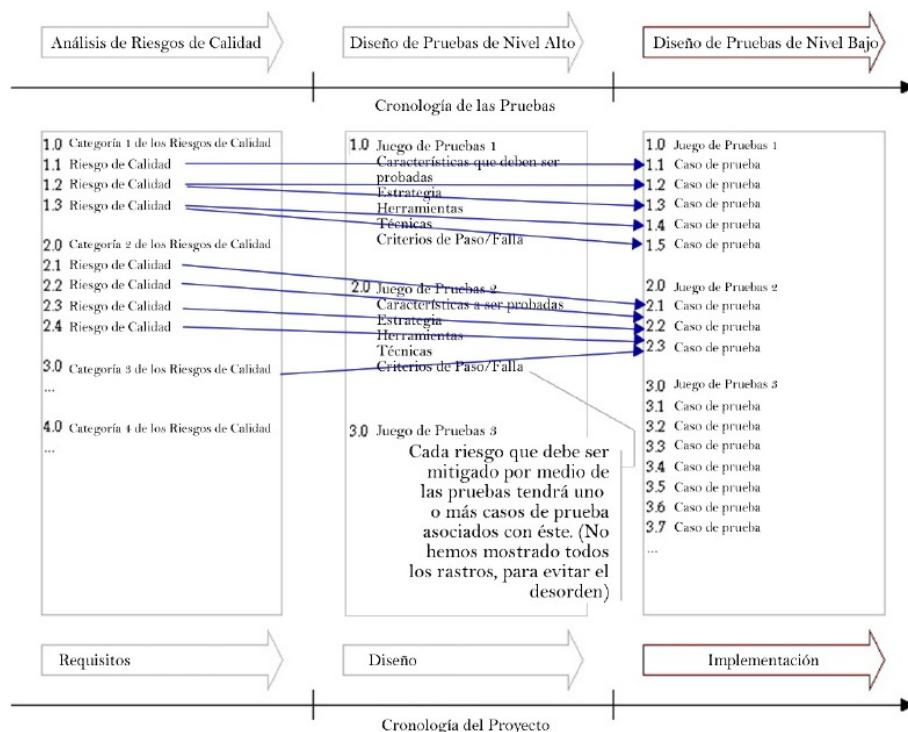


Figura 4.4: Diseño de Pruebas de Nivel Bajo

Por último, examinemos el diseño de pruebas de nivel bajo.

Al igual que antes, en condiciones ideales, el diseño de las pruebas de nivel bajo se produce en paralelo con el ciclo de vida de desarrollo del sistema, en este caso la implementación del sistema. (Por esta razón, en un párrafo anterior, pudimos hablar de versiones tempranas del objeto de pruebas que está disponible durante el diseño de las pruebas de nivel bajo, de tal forma que podamos poner a prueba las pruebas. Este paralelismo es mostrado nuevamente en esta figura como referencia).

En el diseño de las pruebas de nivel bajo, creamos los casos de prueba específicos y finalmente los procedimientos de prueba que cubren las condiciones de las pruebas establecidas en el análisis de las pruebas y el diseño de las pruebas de alto nivel. Como se muestra en esta figura, cada prueba se ajusta a la estructura de los juegos de pruebas, los cuales son simplemente otra vez colecciones lógicas de casos de prueba.

Las flechas de los Riesgos de Calidad hacia los Casos de Prueba muestran la información de la trazabilidad que vamos a capturar, relacionando las pruebas con los riesgos. Cada riesgo que debe ser mitigado por medio de las pruebas, tendrá uno o más casos de prueba asociados a éste. No hemos mostrado todos los trazos en esta figura. Tenga en cuenta que algunos riesgos tienen más de un caso de prueba asociado con esos riesgos. En algunos casos, una prueba se relaciona con

múltiples riesgos. En la siguiente parte de esta sección, usted verá por qué.

Vamos a comenzar con el riesgo. El Glosario del ISTQB se refiere al riesgo como a, "Un factor que podría resultar en futuras consecuencias negativas, generalmente expresado en su impacto y su probabilidad". Bien, eso es un poco difícil de comprender, entonces aquí tiene una definición más fácil de recordar e informal para el riesgo. La posibilidad de un resultado negativo o indeseable.

Cada riesgo tiene algún impacto o daño potencial. Esa es la parte negativa o indeseable. Suele incluir elementos tangibles e intangibles. Por ejemplo, supongamos que la red de cajeros automáticos de un banco se pone inoperable por un día. Hay la pérdida inmediata y tangible de los ingresos que sucederían por las transacciones no procesadas. Hay también el efecto a largo plazo e intangible de la insatisfacción del cliente.

Cada riesgo tiene también alguna probabilidad de convertirse en un resultado. Esto suele ser difícil o imposible de cuantificar, porque no tenemos el mismo tipo de datos actuariales y válidos estadísticamente para los proyectos y los productos de software que las compañías de seguros tienen. Sin embargo, intuitivamente podemos decir que la probabilidad es mayor que 0; las imposibilidades no son riesgos. También, la probabilidad es menor que 1 (o 100%); las certezas no son riesgos tampoco.

Al menos la probabilidad es entre 0 y 1 en el futuro. En otras palabras, cuando estamos hablando de un riesgo en el futuro, podría ocurrir o no. Sin embargo en el pasado el riesgo llegó a ser o no ser un resultado real. Éste es un hecho importante para recordar. Sólo porque un algún resultado posible e incorrecto **no ocurrió** en el pasado no significa que probabilidad es baja, así como el hecho de que algún resultado posible e incorrecto **ocurrió** en el pasado no significa que la probabilidad es alta.

Ahora, en el programa de estudios del ISTQB, hablamos acerca de los riesgos de proyecto y los riesgos de producto. Por el momento, nos enfocaremos en los riesgos de producto. Una definición informal del riesgo de producto que es fácil de recordar es, la posibilidad de que el sistema fallará para satisfacer a los clientes, los usuarios u otros interesados del negocio. Los riesgos de producto se llaman también riesgos de calidad.

Generalmente hablando, los riesgos de calidad o producto caen en algunas categorías amplias. Podríamos entregar software propenso a fallas a los clientes y usuarios. El software, hardware o sistema podría causar potencialmente daños a un individuo o una compañía. El software podría presentar características deficientes en las áreas de por ejemplo funcionalidad, fiabilidad, usabilidad y rendimiento. El software podría tener cuestiones relacionadas a la deficiente integridad y calidad de datos, el cuál puede incluir cuestiones de migración de datos, problemas de conversión de datos, problemas de transporte de datos y violación de los estándares de datos. O el software simplemente no realizaría sus funciones previstas.

Un riesgo de producto o calidad no es un defecto específico. Más bien, es un nombre general para una familia de posibles fallas relacionadas con cualquier número de posibles defectos (o "bugs"). Entonces, suponga que un riesgo de producto es, "El sistema falla en actualizar la interfaz del usuario dentro del medio segundo del recibimiento de una entrada del usuario". Como se puede imaginar, usted podría tener una falla del tiempo de respuesta para cualquiera de las decenas de pantallas que su aplicación podría tener. La falla del tiempo de respuesta podría ser un retraso pequeño pero constante o una pausa intermitente, pero prolongada. El defecto fundamental podría estar en una consulta de la base de datos, una actualización de la base de datos, la latencia de la red, o algún código de procesamiento no elegante.

Éste es un punto importante, porque las pruebas basadas en los riesgos no tienen como objetivo probar cada posible defecto. Tampoco, el análisis de los riesgos tiene como objetivo identificar y evaluar cada posible defecto. En cambio, utilizamos los ítems de riesgo—las condiciones de más alto nivel de las posibles fallas—para enfocar nuestras pruebas. Los varios niveles de riesgo, a través de los ítems de riesgo, nos ayudan a priorizar y a ordenar nuestras pruebas, para poder ocuparnos primeramente con las áreas más riesgosas. Los varios niveles de riesgo nos dicen dónde debemos probar más y dónde debemos probar menos, porque deberíamos invertir más esfuerzo en las áreas más riesgosas. Por medio de las pruebas podemos detectar los problemas que de otra manera afectan a los clientes, y podemos encontrar un porcentaje desproporcionadamente más alto de los problemas más importantes. A medida que probamos, reducimos el nivel de riesgo residual a la calidad del sistema y proporcionamos los resultados basados en los riesgos, informamos al equipo del proyecto y les ayudamos a tomar decisiones acerca de las versiones en base a información fundamentada.

Las pruebas basadas en los riesgos, cuando son realizadas correctamente, comienzan en las etapas del inicio del proyecto. Por medio de un comienzo temprano, éstas proporcionan oportunidades para reducir el riesgo de calidad a través del proyecto. Identificamos los ítems específicos de los riesgos de calidad específicos, evaluamos su nivel de riesgo y utilizamos esa información a través del proceso de pruebas. El conocimiento de los riesgos y sus niveles de riesgo influencian la planificación y el control de las pruebas, la especificación de las pruebas, la preparación de las pruebas y la ejecución de las pruebas.

Los riesgos identificados, junto con su nivel de riesgo asociado, pueden ayudar con muchos

desafíos de las pruebas, así como:

- ¿Cuáles técnicas de pruebas deberíamos utilizar?
- ¿Cuál es el grado de cobertura con el cual deberíamos probar cada área específica?
- ¿Cómo podemos establecer la prioridad y secuencia de nuestras pruebas de tal manera que encontremos los defectos críticos tan temprano como sea posible?
- ¿Cuáles actividades que no pertenecen a las pruebas podrían reducir el riesgo (p.ej., la provisión de entrenamiento para los diseñadores sin experiencia)?

Las pruebas basadas en los riesgos son una actividad de equipo, involucrando a los interesados del negocio a través del proyecto. Algunos interesados del negocio traen experiencia técnica, conocimiento y percepción, mientras otros traen experiencia de negocios, conocimiento y percepción. Esta experiencia colectiva, este conocimiento y esta percepción permiten a los interesados del negocio realizar un trabajo riguroso y exacto de la identificación de los riesgos, la evaluación del nivel de riesgo para cada ítem de riesgo y la determinación de los niveles y el grado de cobertura de las pruebas para abordar aquellos riesgos.

En los métodos más maduros para el desarrollo de software, las pruebas basadas en los riesgos son parte de un método más amplio de gestión de riesgos. Las pruebas basadas en los riesgos ayudan a reducir la probabilidad de falla del producto y dónde queda la probabilidad de falla del producto, para reducir el impacto de los problemas en aquellas áreas. Las pruebas basadas en los riesgos son un método disciplinado, para metódicamente: evaluar (y reevaluar sobre una base regular) lo que pueda ir mal con el software, determinar, utilizando el nivel de riesgo, cuáles riesgos son importantes para ser abordados y las acciones que deben ser implementadas para abordar aquellos riesgos.

Informal.	ISO 9126.	Análisis del Modo de Falla y Efecto.
Empezar con las categorías de riesgos de calidad clásicas.	Empezar con las seis principales características de calidad.	Empezar con categorías, características o subsistemas.
Funcionalidad, estados y transacciones, capacidad y volumen, calidad de datos, manejo de errores y recuperación, rendimiento, estándares y localización, usabilidad, etc.	Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad, Portabilidad (FFUEMP), luego descomponga en subcaracterísticas claves para su sistema.	Los interesados claves del negocio crean una lista de modos de falla posibles, predicen sus efectos en el sistema, en los usuarios, en la sociedad, etc., asignan la severidad, prioridad y probabilidad, luego calculan el número de prioridad de riesgo (RPN).
Establecer la prioridad para probar cada riesgo de calidad con los interesados clave.	Establecer la prioridad para probar cada subcaracterística con los interesados clave.	Los interesados utilizan el RPN para guiar la amplitud y profundidad apropiadas para las pruebas.

Tabla 4.1: ¿Cómo Podemos Analizar los Riesgos de Calidad?

Allá a finales de los años ochenta, cuando entramos en el negocio de las pruebas de software de trabajos anteriores como programador y administrador de sistemas, la Única Manera Verdadera de realizar las pruebas era con una estrategia de pruebas basada en los requisitos. Durante el período de la preparación de las pruebas, un probador serio debía analizar los requisitos, identificar las condiciones de las pruebas, diseñar e implementar las pruebas para cubrir las condiciones de las pruebas y mantener la trazabilidad de las pruebas hacia atrás a los requisitos.

Durante los períodos de la ejecución y los períodos exactos, el probador tenía que ejecutar las pruebas e informar los resultados, incluyendo los informes en cuanto a los requisitos cumplidos y no cumplidos basados en las pruebas.

No es una estrategia deficiente. Sin embargo, es frágil, y tuvimos un montón de problemas con esto. Cuando no recibíamos una buena especificación de los requisitos, las pruebas tendían a reflejar alguno de los problemas en los requisitos, como los, las contradicciones, los comportamientos correctos ambiguos y etc. No nos decía cuánto esfuerzo deberíamos dedicarle a las pruebas de algún requisito dado, excepto cuando recibía requisitos priorizados, lo cual la mayor parte del tiempo no lo recibíamos—y todavía no lo recibimos. Si los requisitos no estaban priorizados, nosotros tampoco sabíamos en qué orden ejecutar las pruebas. Y, por supuesto, si no recibíamos los requisitos en absoluto, nos quedábamos atascados.

En la década de los ochenta, ambos Boris Beizer y Bill Hetzel escribieron, en sus libros, que el riesgo debería conducir las pruebas. Lamentablemente, no dejaron ninguna instrucción acerca de cómo hacerlo.

Entonces, temprano en la mitad de la década de los noventa, frustrados con la fragilidad de las pruebas basadas en los requisitos e inspirado por Beizer y Hetzel, Rex Black decidió encontrar una mejor manera. Él empezó a investigar las formas de analizar los riesgos de calidad. Y así, de forma independiente y en aproximadamente el mismo tiempo, Rick Craig, Paul Gerrard, Félix Redmill, algunos otros, y él desarrollaron métodos muy similares para las pruebas basadas en los riesgos. A continuación vamos a describir éste método.

Siendo una persona quien preferiría adaptar en vez de inventar, buscó en las técnicas de los años 90 en el mercado de las ideas. Afortunadamente, él tenía mucha experiencia con los sistemas hardware/software, así que se encontró con el concepto del Análisis de Modos de Fallas y Efectos. En esta técnica, usted utiliza un framework con categorías, características de calidad (a la ISO 9126), o subsistemas. Usted luego trabaja con los interesados del negocio clave para identificar los posibles modos de falla, predecir sus efectos en el sistema, el usuario, la sociedad, etc. Basado en estos efectos, usted asigna la severidad, la prioridad y la probabilidad. Usted calcula el número de prioridad del riesgo (RPN) como el producto de estos tres valores.

Esto fue una revelación para él. El Análisis de Modos de Falla y Efectos resolvieron sus problemas con las pruebas basadas en los requisitos. Porque esto dependió de la lluvia de ideas de grupo o entrevistas de uno-a-uno, esto era inmune a que si recibió o no los requisitos. (Por supuesto, si él hubiera recibido los requisitos, podía haberlos utilizado como una entrada, pero no estaba comprometido con otro equipo para que fueran escritos). El número de prioridad del riesgo le decía la cantidad del esfuerzo a dedicarle a la preparación y la realización de las pruebas relacionadas con cada ítem de riesgo. El orden de la prioridad del riesgo también le decía el orden en el cual debería ejecutar las pruebas.

Lamentablemente, para la mayoría de nuestros clientes, esto no funcionó demasiado bien. El problema con el Análisis de Modos de Falla y Efectos es que se genera demasiada información. Miles de modos de falla, en muchos casos.

Así, a lo largo de los años, hemos desarrollado un método ligero e informal que funcionará en casi cualquier situación, aun conservando las principales ventajas del Análisis de Modos de Falla y Efectos. Hemos desarrollado una lista de categorías clásicas de los riesgos de calidad, así como la funcionalidad, los estados y las transacciones, la capacidad y el volumen, la calidad de datos, el control de errores y la recuperación después de fallas, el rendimiento, los estándares y la localización, la usabilidad, etc. Trabajando con los interesados clave del negocio, identificamos los riesgos de calidad, luego establecemos la prioridad para probar cada riesgo de calidad.

Si quiere algo un poco más formal, puede utilizar el ISO 9126 para estructurar su análisis de los riesgos. Usted comienza con las seis principales características de calidad, Funcionalidad, Fiabilidad, Usabilidad, Eficiencia, Mantenibilidad y Portabilidad. Luego, las divide en subcaracterísticas clave que aplican para su sistema, siguiendo el estándar ISO 9126. Por último, identifica los ítems de los riesgos de calidad para cada subcaracterística y establece la prioridad para probar cada riesgo, nuevamente trabajando con los interesados clave del negocio.

A partir de la página siguiente, vamos a explicar la técnica informal. Sin embargo tenga en cuenta que independientemente de la técnica, los aspectos más importantes son la participación multifuncional de los interesados del negocio, el consenso, y una perspectiva del mejor resultado posible.

Los riesgos de calidad son potenciales problemas de sistemas que pueden reducir la satisfacción del usuario

	El Número de prioridad de riesgo: Medida global del riesgo del problema	Riesgo de negocio (operacional): Impacto del problema	Riesgo técnico: Probabilidad del problema	El seguimiento de la información de requisitos, diseño, u otras bases de riesgo	
Riesgo de Calidad	Riesgo Tec.	Riesgo Neg.	Riesgo Pri. #	Alcance de las pruebas	Seguim.
Categoría de Riesgo 1					
Riesgo 1					
Riesgo 2					
Riesgo n	Una jerarquía de categorías de riesgo puede ayudar a organizar la lista y estimular su memoria 1 = Muy alto 2 = Alto 3 = Mediano 4 = Bajo 5 = Muy bajo	El producto del riesgo técnico y riesgo de negocio, desde 1-25.	1-5 = Extenso 6-10 = Amplio 11-15 = Superficial 16-20 = Oportunidad 21-25 = Defectos del Informe		

Figura 4.5: Riegos de Calidad

En esta figura, usted puede observar una plantilla que puede ser utilizada para capturar la información que usted identifica en el análisis de los riesgos de calidad.

El proceso del análisis de los riesgos de calidad consiste en las siguientes actividades. En primer lugar, identificará los riesgos de calidad y luego evaluará sus niveles de riesgo. Basado en el nivel de riesgo, determinará la prioridad general de las pruebas y el alcance de las pruebas. Por último, si los riesgos surgen de los requisitos específicos o elementos de la especificación del diseño, establezca la trazabilidad hacia atrás a estos ítems. Examinemos estas actividades y cómo ellas generan la información para llenar esta plantilla.

Primero, recuerde que los riesgos de calidad son problemas potenciales del sistema los cuales podrían reducir la satisfacción del usuario.

Podemos utilizar una jerarquía de las categorías de los riesgos para organizar la lista y para refrescar su memoria. Al trabajar con los interesados del negocio, identificamos uno o más riesgos de calidad en cada categoría y llenamos la plantilla.

Una vez que los riesgos han sido identificados, ahora podemos volver atrás y evaluar el nivel de riesgo, porque podemos ver los ítems de riesgo en relación con los demás. Usualmente, utilizamos dos factores principales para la evaluación del riesgo.

La primera es la probabilidad del problema, que es influenciada en su mayor parte por las consideraciones técnicas. Entonces, a veces lo llamamos "riesgo técnico" para recordarnos de ese hecho.

El segundo es el impacto del problema, que está influenciado la mayor parte por las consideraciones de negocios u operaciones. Entonces, a veces lo llamamos "riesgo de negocios" para recordarnos de ese hecho.

Tanto la probabilidad como el impacto pueden ser evaluados en una escala ordinal así como alto, medio y bajo. Nosotros preferimos utilizar una escala de cinco puntos, de muy alto a muy bajo.

Teniendo en cuenta la probabilidad y el impacto, necesitamos una sola medida conjunta del riesgo para el ítem de riesgo de calidad. Utilizamos la frase número de prioridad de riesgo para esto, aunque no es exactamente el mismo término de FMEA (Análisis de los Modos de Fallas y los Efectos). Para crear el número de prioridad de riesgo, combinamos la probabilidad y el impacto. Una forma es traducir la escala ordinal en una escala numérica, por ejemplo:

1 = Muy alto.

2 = Alto.

3 = Medio.

4 = Bajo.

5 = Muy bajo.

Entonces podemos calcular el número de prioridad de riesgo como el producto de los dos números. Es más bien poco elegante, pero funciona. Siéntase libre de elaborar sus propias fórmulas para el cálculo de este número si la simple multiplicación no funciona.

Ahora, el número de prioridad de riesgo puede ser utilizado para la secuencia de las pruebas. Sin embargo, todavía necesitamos determinar el alcance de las pruebas. Una forma de hacerlo es de dividir el número de prioridad de riesgo en cinco grupos y utilizarlos para determinar el esfuerzo de la prueba:

1-5 = Extenso.

6-10 = Amplio.

11-15 = Superficial.

16-20 = Oportunidad.

21-25 = Informar de defectos solamente.

Otra vez, siéntase libre para afinar estos grupos para que coincida con sus necesidades.

Mientras pasa por el proceso del análisis de los riesgos de calidad, es probable que usted genere varios subproductos útiles. Estos incluyen los supuestos de la implementación que usted y los interesados del negocio hicieron acerca del sistema en la evaluación de la probabilidad. Usted querrá validar estos, y ellos podrían demostrar sugerencias útiles. Los subproductos incluyen también los riesgos del proyecto que usted descubrió, los cuales el director del proyecto puede abordar. Tal vez lo más importante es que los subproductos incluyen los problemas con los requisitos, el diseño u otros documentos de entrada. Ahora podemos evitar que estos problemas se conviertan en defectos reales del sistema. Observe que los tres habilitan el rol preventivo de los defectos de las pruebas tratados anteriormente en este libro.

Este proceso no es trivial de llevar a cabo, pero es tremadamente poderoso, cuando es realizado correctamente. Permitanos darle algunas pautas finales acerca de este proceso para ayudarle a empezar.

En primer lugar, y lo más importante es, de asegurar el empleo de un equipo multifuncional de lluvia de ideas. El análisis de los riesgos de calidad no crea ninguna sabiduría, sólo la extrae de los interesados del negocio. Mientras usted más perspectivas incluya, mejor informado éste será.

Segundo, utilice un proceso de dos etapas: identifique los ítems de riesgo, a continuación evalúe el nivel de riesgo. Los riesgos están relacionados entre sí, por lo tanto no puede asignar el nivel de riesgo hasta que haya visto todos los riesgos.

Tercero, sea tan específico como necesite serlo, pero no más específico. Es decir, si está considerando un ítem de riesgo y si este debería ser dividido en dos o más ítems de riesgo, pregúntese: "¿Me ayudaría a distinguir entre los diferentes niveles de riesgo la separación de estos ítems de riesgo en dos o más ítems de riesgo?" Si la respuesta es no, no los divida.

Cuarto, asegúrese de considerar el riesgo técnico y de negocios. Es tan fácil de caer en la trampa, como lo han hecho muchos en este negocio, pensando en las pruebas como alguna especie de cacería fanática de defectos. Encontrar defectos es divertido, pero también es importante construir la confianza y gestionar los riesgos. Si deja que la probabilidad dirija su proceso de pruebas, se convertiría en un fanático de la caza de defectos. Pero si también considera el impacto, entonces usted podrá realmente ayudar a su equipo a gestionar el riesgo y construir la confianza.

Por último, entienda que este proceso es falible. Usted cometerá errores, en algunos casos debido a la información limitada. Eso está bien, siempre y cuando haga el seguimiento y vuelva a alinear el análisis de los riesgos con las pruebas y el proyecto en hitos clave del proyecto.

Examinemos algunas plantillas de documentación que puede utilizar para capturar la información a medida que analiza, diseña e implementa sus pruebas. La primera es la Especificación del Diseño de Pruebas IEEE 829.

La especificación del diseño de pruebas describe una colección de casos de prueba y las condiciones de pruebas que abarcan en un nivel alto. Esta plantilla incluye las siguientes secciones:

- Identificador de la especificación del diseño de pruebas.
- Características que deben ser probadas (en este juego de pruebas).
- Refinamientos del método (técnicas específicas, herramientas, etc.).
- Identificación de las pruebas (trazabilidad de los casos de prueba en el juego de pruebas).
- Criterios de paso/falla de las características (p.ej., oráculo de pruebas, base de pruebas, sistemas heredados, etc.).

La colección indicada de casos de prueba en la especificación del diseño de pruebas es a menudo llamada un juego de pruebas.

La secuencia de los juegos y casos de prueba dentro de los juegos de prueba es a menudo dirigida por la prioridad del riesgo y el negocio. Por supuesto, la secuencia debe ser afectada por las restricciones, los recursos y el progreso del proyecto.

Luego viene la Especificación del Caso de Prueba IEEE 829. Una especificación del caso de prueba describe los detalles de un caso de prueba. Esta plantilla incluye las siguientes secciones:

- Identificador de la especificación del caso de prueba.

- Ítems de pruebas (lo que tiene que ser entregado y probado).
- Especificaciones de las entradas (entradas del usuario, archivos, etc.).
- Especificaciones de las salidas (resultados esperados, incluyendo pantallas, archivos, tiempo, etc.).
- Necesidades del entorno (hardware, software, personas, accesorios...).
- Requisitos especiales de procedimiento (intervención del operador, permisos, etc.).
- Dependencias entre casos (si es necesario para establecer condiciones previas).

Si bien esta plantilla define un estándar para los contenidos, la pregunta de lo que es un caso de prueba es ciertamente una pregunta abierta. En la práctica, los casos de prueba varían significativamente en esfuerzo, duración y número de condiciones de pruebas cubiertas. Finalmente viene la Especificación del Procedimiento de Prueba IEEE 829. Una especificación del procedimiento de prueba que describe cómo ejecutar uno o más casos de prueba. Esta plantilla incluye las siguientes secciones:

- Identificador de la especificación del procedimiento de prueba.
- Propósito (p.ej., cuáles pruebas son ejecutadas).
- Requisitos especiales (habilidades, permisos, entorno, etc.).
- Pasos del procedimiento (registrar, instalar, iniciar, proceder [los pasos mismos que deben ser ejecutados], medición de resultados, apagar/suspender, reiniciar [si es necesario], detener, concluir/eliminar, contingencias).

Mientras que el estándar IEEE 829 distingue entre los procedimientos de prueba y casos de prueba, en la práctica los procedimientos de prueba son a menudo embebidos en los casos de prueba.

Un procedimiento de prueba es referido a veces como un guión de prueba. Un guión de prueba puede ser manual o automatizado.

Glosario del ISTQB

Guón de prueba: Comúnmente utilizado para referirse a una especificación del procedimiento de prueba, especialmente uno automatizado.

Para terminar esta sección permítanos explicar el concepto de la trazabilidad o la cobertura de los casos de prueba un poco más.

El glosario define la trazabilidad como “La habilidad de identificar los ítems relacionados en la documentación y el software, así como los requisitos con las pruebas asociadas”. Note que es otra manera de decir que estamos asociando la prueba con la base de pruebas, lo cual es “la documentación en la cual los casos de prueba se basan”.

La cobertura es un concepto similar, relacionado con la medición de la relación y se la define como “el grado en el cual un ítem específico de la cobertura ha sido ejercido por un juego de pruebas”, donde un ítem de la cobertura puede ser cualquier cosa en la que estemos interesados en cubrir con las pruebas.

La manera simple de pensar acerca de la trazabilidad es que estamos midiendo o correlacionando nuestras pruebas contra las áreas de interés en el producto. En otras palabras, cómo se relaciona lo que **estamos** probando con lo que nos **proponemos** que debe ser probado. Esta medición nos dice el grado de cobertura en la cual nuestras pruebas están alineadas con lo que necesitamos cubrir. En las pruebas basadas en los riesgos, las mediciones de la cobertura nos dicen si estamos tratando los riesgos correctamente. Podemos medir dónde es la cobertura muy poca o demasiada. Luego podemos mejorar las pruebas para llenar los vacíos y hacer el esfuerzo más eficiente.

Hay muchas maneras de medir la cobertura en el uso común. Para las estrategias basadas en los requisitos, planificariamos medir principalmente contra las especificaciones de los requisitos. Para las estrategias de pruebas más amplias, podríamos también medir contra las especificaciones del diseño.

Para las estrategias basadas en los riesgos, medimos principalmente contra los riesgos, aunque también tenemos la trazabilidad de los riesgos hacia atrás a las especificaciones de los requisitos y el diseño. Cuando la interoperabilidad es un tipo de pruebas importante, también podría medir la cobertura de las configuraciones, las interfaces y otros también.

Existe también la cobertura de código, que examinaremos después en este capítulo.

				Caso de Prueba										
	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	2.1	2.2	2.3	2.4	2.5	Total
Espec.														
1.1	1	1		1	1		2							6
1.2	2			2	1				1	2				8
1.3														0
1.4		2								2				4
1.5	1			1				1	1					4
1.6					2									2
1.7			2		1		2		2					7
1.8					2						1			3
1.9	1			1						2				4
1.10		1					1	1		1				4
Total	5	4	2	5	4	3	3	4	2	2	7	1	0	

Figura 4.6: Cobertura de Casos de Pruebas (Trazabilidad)

Hemos ilustrado una técnica para capturar la cobertura en esta figura. Podemos utilizar una hoja de cálculo para mostrar el caso de prueba horizontalmente a través de la parte superior y la especificación de los requisitos verticalmente en el lado izquierdo hacia abajo. Cada celda contiene una medida del grado de cobertura que un caso de prueba dado proporciona para un elemento de especificación dado.

El espacio en blanco o 0 significa sin cobertura, 1 significa una cobertura indirecta, en el sentido de que solo aborda periféricamente el elemento, 2 significa una cobertura directa, en el sentido de que la prueba es diseñada para decirnos, si este área en particular tiene problemas. El total nos ayuda a detectar los desalineamientos de las pruebas con el énfasis apropiado del conjunto de las pruebas para detectar vacíos en las pruebas.

Sin embargo la realidad es, que usted no lo haría probablemente de esta manera. Para un sistema grande, puede tener cientos de elementos en su base de pruebas, y cientos de casos de prueba. Eso resulta en una hoja de cálculo con diez miles, algunas veces cientos de miles de celdas que hacen el seguimiento. Introduciendo la información errónea en tal hoja de cálculo es muy posible y es una pesadilla de mantener. El mejor método es utilizar una base de datos. Esas bases de datos son la parte esencial de las herramientas de gestión de pruebas que pueden capturar la información de la trazabilidad.

4.1.1 Ejercicios

Ejercicio 1

Basado en su lectura del Documento de los Requisitos de Marketing de Omninet, el Documento de los Requisitos del Sistema Omninet y su experiencia con las pruebas y los defectos, realice un análisis de los riesgos para Omninet.²³

Argumente.

Solución del Ejercicio 1

Al realizar este análisis de los riesgos de calidad, usted debería haber recordado que los riesgos se pueden relacionar con los quioscos, el centro de llamadas y el centro de datos. Algunas personas tienden a olvidar todo lo que no tiene que ver con la funcionalidad de cara al cliente, lo cual conduce a muchos problemas pasados por alto.

Otra equivocación común que la gente comete en el análisis de los riesgos es olvidar que los riesgos no sólo se relacionan con la funcionalidad, sino también con el rendimiento, la seguridad, la privacidad y otras características de calidad no funcionales.

Al realizar el análisis de los riesgos, usted tuvo que tener cuidado de mantener todos los riesgos en la perspectiva mientras se asignaron los niveles de riesgo. De otra manera habría tenido una inflación de riesgos, donde todo sea calificado como de riesgo alto.

También debería haber estado listo para hacer algunas suposiciones razonables de la implementación. Porque no todos los probadores son expertos técnicos, incluyendo los tecnólogos

en el proceso del análisis de los riesgos, quienes puedan llenar estos espacios en blanco es esencial.

Para revisar el proceso, primero debería haber identificado los riesgos, después priorizarlos. Una vez que tuviera una prioridad agregada, debería haber utilizado esa prioridad para guiar el grado de cobertura de las pruebas en cada área de riesgo. En el libro sugerimos que utilice cinco niveles del grado de cobertura:

- Pruebas extensivas. Probar el área entera del riesgo, con muchas variaciones.
- Pruebas amplias. Probar el área entera del riesgo, pero con pocas variaciones.
- Pruebas superficiales: Probar una muestra del área del riesgo, explorándola brevemente.
- Pruebas de oportunidad. Probar el área del riesgo sólo si alguna otra prueba lo lleva a usted al área.
- Informar defectos. Si ve problemas en esta área del riesgo, infórmelos, pero no haga nada más.

A medida que trabajó a través del Documentos de los Requisitos de Marketing de Omninet y el Documento de los Requisitos del Sistema Omninet, descubriendo los riesgos asociados con cada área, debería haber retenido la información de la trazabilidad entre la sección de los documentos y los riesgos asociados. Esta información de la trazabilidad es útil para el diseño de pruebas y la creación de los informes de resultados.

En la tabla 4.2, proporcionamos nuestra solución para el ejercicio del análisis de los riesgos de calidad. En el material que sigue a la tabla, damos algunas observaciones acerca de nuestra solución así como también algunos hallazgos interesantes que ocurrieron cuando pasamos por el proceso del análisis de los riesgos de calidad.

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
1.000	<i>Funcionalidad</i>					
1.001	Los clientes no pueden acceder a la Web (en absoluto).	5	1	5	Extensivas	3.1
1.002	La contabilidad del tiempo (periodos, límites) manejada incorrectamente.	4	2	8	Amplias	3.1.2 3.1.7
1.003	El tiempo quedándose demasiado rápido (engañoso al usuario) o demasiado lento (engañoso a Omninet).	4	3	12	Superficiales	3.1.2 3.1.7
1.004	Pago válido rechazado /Pago inválido aceptado.	5	2	10	Amplias	3.1.2
1.005	Proceso de expiración gestionado incorrectamente	3	2	6	Amplias	3.1.2
1.006	Los navegadores alternativos especificados no son compatibles.	4	4	16	De oportunidad	3.1.3
1.007	Contenido inapropiado no es bloqueado apropiadamente.	1	1	1	Extensivas	3.1.6
1.008	Contenido apropiado bloqueado inapropiadamente.	1	1	1	Extensivas	3.1.6

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
1.009	El software de cortafuegos/antivirus no bloquea los gusanos, virus, etc.	1	1	1	Extensivas	3.1.6
1.010	El cierre de sesión del usuario falla.	5	2	10	Oportunidad	3.1.7
1.011	El cierre de sesión del usuario (antes o en la expiración) resulta en un reembolso.	3	2	6	De oportunidad	3.1.7
1.012	La confidencialidad no es preservada.	2	2	4	Extensivas	3.1.8
1.013	El navegador no sale al final de la sesión.	5	2	10	Amplias	3.1.8
1.014	El navegador no se reinicia después de salir al final de la sesión.	4	2	8	Amplias	3.18
1.015	Los agentes del centro de llamadas no pueden acceder/controlar las sesiones actuales.	2	2	4	Extensivas	3.1 3.2.2
1.016	Los agentes del centro de llamadas no pueden acceder a la información acerca de las sesiones pasadas.	3	2	6	Extensivas	3.1 3.2.2
1.017	El quiosco no informa el estatus (cada hora/en absoluto) al centro de llamadas.	3	2	6	Amplias	3.2.2
1.018	El centro de llamadas no se puede conectar al quiosco para reunir el estado.	3	2	6	Amplias	3.2.2

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
1.019	El agente del centro de llamadas no puede modificar la sesión del usuario.	2	5	10	Superficiales	3.2.4
1.020	El agente del centro de llamadas puede dar más de 60 minutos de tiempo.	3	3	9	Amplias	3.2.4
1.021	El agente del centro de llamadas no puede terminar la sesión.	2	1	2	Extensivas	3.2.5
1.022	El reembolso por la terminación es excesivo/insuficiente.	2	3	6	Amplias	3.2.5
2.000	Localización					
2.001	El quiosco no está configurado en el idioma local principal.	5	2	10	Amplias	3.1.5
2.002	No son accesibles todos los idiomas apoyados por el navegador o sistema operativo.	3	5	15	Superficiales	3.1.5
2.003	El bloqueo del sitio ofensivo no es localizado correctamente.	1	2	2	Extensivas	3.1.5 3.1.6

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
2.004	Las monedas locales no son correctamente tratadas.	5	2	10	Amplias	3.1.2
2.005	Zonas horarias locales causan problemas de actualización y fiabilidad.	3	2	6	Amplias	3.1.5
3.000	Usabilidad e Interfaz de Usuario					
3.001	La pantalla de Bienvenida no es atractiva.	1	1	1	Extensivas	3.1.1
3.002	Las pantallas de pago/procesar el desaliento/ahuyentar a los clientes potenciales	1	1	1	Extensivas	3.1.2
3.003	Las pantallas de expiración/ procesar el desaliento/ahuyentar a los clientes repetidos.	1	1	1	Extensivas	3.1.2
3.004	Proceso de selección de idioma confuso e inutilizable.	1	2	2	Extensivas	3.1.5
3.005	Mensaje de terminación inapropiado.	3	2	6	Amplias	3.2.5
3.006	Esquemas de colores, tipos de letra, etc. no apropiados/accesibles.	1	1	1	Amplias	3.1

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
4.000	<i>Fiabilidad</i>					
4.001	La granja de servidores no puede soportar 1.000 quioscos.	1	1	1	Extensivas	2
4.002	Las sesiones del quiosco se caen durante la utilización.	1	1	1	Extensivas	3.1
4.003	Los quioscos no pueden muy frecuentemente acceder la Web.	1	1	1	Extensivas	3.1
4.004	Los quioscos con dial-up (PSTN) establecen algunas veces conexión de <= 50 KBPS.	1	2	2	Extensivas	3.1.4
4.005	Los quioscos con cable/DSL establecen algunas veces conexión de <= 128KBPS.	4	2	8	Amplias	3.1.4
4.006	Algunas veces falla el acceso/control/terminación del agente del centro de llamadas.	2	3	6	Amplias	3.2.1-3.2.5
4.007	La disponibilidad del centro de llamadas demasiado baja.	3	1	3	Extensivas	3.2

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
4.008	La disponibilidad de los servidores de enlace es demasiado baja.	3	1	3	Extensivas	3.1
4.009	Los quioscos no pueden recuperarse de una falla de energía u otra falla externa.	5	1	5	Amplias	3.1
4.010	Problemas de compatibilidad del sitio Web/URL, algunos sitios son mostrados incorrectamente.	2	4	8	De oportunidad	3.1
4.011	Intermitente rechazo de pagos válidos/aceptación de pagos inválidos.	5	4	20	Informe de Defectos	3.1.2
5.000	<i>Rendimiento</i>					
5.001	Proceso de pago inicial o de expiración demasiado lento.	5	2	10	Superficiales	3.1.2
5.002	Los quioscos con Dial-Up (PSTN) nunca establecen conexión de > 50 KBPS.	5	2	10	Amplias	3.1.4
5.003	Los quioscos con Cable/DSL nunca establecen conexión de > 128 KBPS.	5	2	10	Amplias	3.1.4

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
5.004	El acceso/control/terminación de los agentes del centro de llamadas demasiado lento. (después del hecho)	2	1	2	Extensivas	3.2.1-3.2.5
5.005	Problemas de rendimiento de actualización/sesión.	4	3	12	Superficiales	3.2.1
6.000	<i>Sostenibilidad("Supportability")</i>					
6.001	El software de antivirus/cortafuegos no es actualizado regularmente.	5	3	15	Superficiales	3.1.6
6.002	Las actualizaciones de software no ocurren automáticamente.	4	1	4	Extensivas	3.2.1
6.003	Las actualizaciones de software nunca tienen éxito.	4	1	4	Extensivas	3.2.1
6.004	Las actualizaciones de software fallan demasiado frecuente.	5	1	5	Extensivas	3.2.1
6.005	El quiosco permanece conectado después de comprobar o realizar la actualización.	5	4	20	De oportunidad	3.2.1

No.	Riesgo de Calidad	Riesgo Téc.	Riesgo de Neg.	Pri. # de Riesgo	Grado de Cobertura de las Pruebas	Trazabilidad
6.006	El proceso de reintento en la sobrecarga falla (no se desconecta/no se vuelve a conectar)	3	1	3	Extensivas	3.2.1
6.007	Los agentes no pueden hacer las actualizaciones a los quioscos.	3	1	3	Extensivas	3.2.1

Tabla 4.2: Solución-Análisis de los Riesgos de Calidad de Omninet.

Trabajando directamente de las especificaciones realizadas que sencillamente identifican los riesgos, usted *puede* realizar los análisis de los riesgos sin especificaciones de ninguna clase. Lo hemos hecho exitosamente. Sin embargo, es más fácil cuando tiene algo escrito, aún un documento

imperfecto como el Documento de los Requisitos de Marketing de Omninet.

Para algunos riesgos, hemos desviado el grado de cobertura de las pruebas de lo que el número total de prioridad de riesgo sugeriría. Por ejemplo, hemos asignado las pruebas de oportunidad para "el cierre de sesión del usuario falla", la cual tenía un número de prioridad de riesgo 10. ¿Por qué? Porque es una tarea de programación simple, lo más probable, y puede realizar fácilmente su prueba de integración en algunas otras pruebas.

¿Puede detectar los otros riesgos para los cuales hemos desviado el grado de cobertura de las pruebas? ¿Puede darse cuenta por qué? La documentación de esas decisiones y las suposiciones es probablemente lo mejor. Sin una explicación como la que está en el párrafo de arriba, es difícil de saber nuestro razonamiento.

Note que la relación entre los riesgos de calidad y los elementos de la especificación de los requisitos es de muchos-a-muchos. Un requisito se puede relacionar con múltiples riesgos. Eso está bien. Un riesgo se puede relacionar con múltiples requisitos. Eso está bien, también, en números pequeños. Sin embargo, si encuentra que tiene demasiados requisitos asociados con un riesgo, usted probablemente no está siendo lo suficientemente específico. Por ejemplo, "el sistema no funciona" se relacionaría a cualquier requisito, pero no le ayuda a darse cuenta a dónde enfocar sus pruebas.

A medida que priorizamos los riesgos, nos dimos cuenta que algunos ítems que habíamos incluido, como dos riesgos, se plegaron en un sólo riesgo. Por ejemplo, "Los agentes del centro de llamadas no pueden acceder/controlar las sesiones actuales", eran originalmente dos ítems en una línea cada uno, uno relacionado con el acceso y el otro con el control. Cuando vimos que tenían los mismos números de riesgo de prioridad, entonces los combinamos. Tomamos ventaja de esos descubrimientos para mantener las tablas del análisis de los riesgos tan cortas como era posible.

Porque solamente realizamos este análisis, estamos seguros que pasamos por alto algunos riesgos importantes. Los equipos multidisciplinarios que trabajan juntos no tienden a pasar por alto tantos riesgos.

Probablemente también obtuvimos los números de prioridad equivocados en muchos casos. Por supuesto, la prioridad técnica y de negocios, ambas dependen un poco del contexto técnico y los negocios del proyecto. Al trabajar en un proyecto hipotético como éste, es difícil decir si las prioridades están correctas o incorrectas.

Examinando las prioridades y la cobertura de las pruebas, puede observar cómo la correlación de la prioridad total con la cobertura de las pruebas que utilizamos aquí, resulta en una tendencia hacia las pruebas extensivas. De nuevo, siéntase libre de adoptar otro esquema de mapeo para cambiar la parcialidad hacia las pruebas amplias o incluso pruebas superficiales si es apropiado.

Ahora, además de encontrar un número de riesgos interesantes de calidad que atender durante las pruebas, también encontramos una serie de cuestiones relacionadas con la especificación de requisitos en particular y con el proyecto en general:

1. No todos los acrónimos definidos en la sección 1.1 son realmente utilizados en el documento. Esto es confuso, porque no estamos seguros por qué necesitamos saber acerca de ellos.
2. Sería útil tener esos prototipos de pantallas mencionados en la sección 1.2 para el análisis de los riesgos.
3. En la sección 2, se menciona "el tercer cuartal financiero", ¿Pero de qué año?
4. En la sección 3.1.3, se le ofrece al usuario la elección de navegadores. ¿No debería decir esto, "navegadores alternativos"? Asumiríamos que el programa entero del quiosco sería una aplicación web corriendo algún tipo de servidor Web local (p.ej. IIS o Apache), lo cual significaría que, excepto en los períodos de iniciación o como un resultado de un error fatal, el software del quiosco se ejecutará siempre dentro en un navegador. Esta función parecería que proporciona el cambio a otros navegadores alternativos.
5. Si el comentario de arriba es correcto, entonces la falta de especificación de navegadores por defecto versus alternativos para todos los sistemas operativos compatibles de los quioscos es una omisión en los requisitos o en lo mínimo debería ser atendido en la especificación del diseño.
6. En referencia a la sección 3.1.2, ¿Qué ocurre si el tiempo expira mientras el usuario está comprando un período de tiempo adicional? ¿Termina la sesión del usuario, llevándose consigo todos los datos de la sesión? (como está descrito en la sección 3.1.8).
7. Siguiendo con el comentario de arriba y de nuevo refiriéndose a la sección 3.1.2, ¿Cuánto es un período de tiempo de espera razonable para el proceso del pago inicial o de la compra de tiempo adicional? ¿Qué pasa cuando el usuario cambia de idea y se retira del quiosco?
8. Siguiendo esta línea más allá y de nuevo refiriéndose a la sección 3.1.2, ¿Tiene que volver a pasar su tarjeta un cliente con tarjeta de crédito o débito para comprar tiempo adicional? Si no, considere los riesgos asociados con un cliente que se retira del quiosco sin terminar su sesión.
9. Con respecto a la sección 3.1.5, el "idioma local principal" podría variar de un lado del pueblo al otro. Por ejemplo, en San Antonio, Texas, donde vive Rex Black, cerca de la mitad de la población es hispánica, y español es el idioma principal para mucha gente en ciertos barrios.

- Los estudios demográficos de la ubicación real del quiosco—más que solo datos del censo para una ciudad o región—tendrían que dirigir esta decisión para evitar quioscos inutilizados.
10. De nuevo con respecto a la sección 3.1.5, identificamos el “proceso de la selección confusa e inutilizable del idioma” como un riesgo. Aquí hay un ejemplo. Rex Black tenía un teléfono que un cliente le dejó utilizarlo mientras está en Israel. El software en el teléfono es compatible con múltiples idiomas, incluyendo el inglés. Sin embargo, mientras está en modo hebreo, la única manera de encontrar el modo de selección del inglés es —lo adivinó— en hebreo. Porque el habla muy poco hebreo, tuvo que pedirle a un colega que habla hebreo que reconfigure el teléfono.
 11. Con respecto a la sección 3.2.1, identificamos, que “las actualizaciones de software fallan demasiado frecuente”, pero por supuesto tendríamos que definir—ya sea idealmente en la especificación del diseño o por lo menos en los mismos casos de prueba— lo que significa “demasiado frecuente”. Claramente el 100% de la fiabilidad del proceso de la actualización no es razonable, pero ¿Cuál es una tasa razonable de falla?
 12. Con respecto a la sección 3.2.2, los agentes del centro de llamadas tendrán acceso a información sensible de los clientes como números de tarjetas de crédito, direcciones de internet visitadas—URLs—, etc. Éste no es un riesgo de prueba, sino más bien un riesgo de negocios: ¿Cómo nos aseguramos que esta gente no haga uso incorrecto de esta información? Tal vez el sistema debería deshabilitar la captura de pantallas, el guardado de archivos y los correos electrónicos de las computadoras de escritorio de los agentes del centro de llamadas. Si es así, entonces eso *tendría* implicaciones en las pruebas.
 13. Con respecto a la sección 3.2, identificamos la “la disponibilidad del centro de llamadas demasiado baja”, pero tendríamos que definir—ya sea idealmente en la especificación del diseño o por lo menos en los mismos casos de prueba— lo que significaría baja disponibilidad. Claramente el 100% de la disponibilidad del centro de llamadas no es razonable, pero ¿Cuánto es la cantidad razonable de tiempo de inactividad anual?
 14. Con respecto a la sección 3.1, identificamos que “la disponibilidad de los servidores de los puertos de enlace es demasiada baja”, pero tendríamos que definir—ya sea idealmente en la especificación del diseño o por lo menos en los mismos casos de prueba— lo que significaría baja disponibilidad. Claramente el 100% de la disponibilidad del centro de llamadas no es razonable, pero ¿Cuánto es la cantidad razonable de tiempo de inactividad anual?
 15. Con respecto a la sección 3.1, identificamos que “los quioscos no pueden acceder a la web demasiado frecuente”, pero tendríamos que definir—ya sea idealmente en la especificación del diseño o por lo menos en los mismos casos de prueba— lo que significaría “demasiado frecuente”. Claramente el 100% de la disponibilidad no es razonable, pero ¿Cuánto es la cantidad razonable de tiempo de inactividad anual?
 16. Con respecto a las secciones 3.2.1 a la 3.2.5, identificamos “el acceso/el control/la terminación del agente del centro de llamadas demasiado lento (después del hecho)” como un riesgo. Lo que queremos decir es que podría ser el caso de que el daño ya ha sido hecho en el momento que el agente del centro de llamadas se dé cuenta, tratará de afectar o tratará de terminar algún tipo de uso antisocial del quiosco o funcionalidad incorrecta del quiosco relacionada con el software (p.ej. la infección por un gusano). Esto requiere de algún razonamiento más cuidadoso por parte de los diseñadores del sistema, idealmente los expertos en seguridad.
 17. Con respecto a la sección 3.1.5 y 3.1.6, identificamos un riesgo como por ejemplo “el sitio ofensivo que bloquea no es correctamente localizado”. Las costumbres locales, las religiones, etc. influencian lo que es ofensivo. Los diseñadores y los programadores deben ser cuidadosos para tomar en cuenta esto.
 18. Con respecto a la sección 3.1, ¿Deberían entrar los quioscos en un modo de ahorro de energía cuando el espacio público (en el cual residen) es cerrado? Si es así, entonces hay implicaciones de las pruebas.
 19. No entramos en muchos detalles con respecto a la sección 3.1.2, el proceso de pago. Hay muchas cosas que pueden salir mal aquí. Estamos asumiendo que los diseñadores serían inteligentes y utilizarían un subsistema de pago comercial de distribución masiva, así como aquellos encontrados en los quioscos de parqueos y los lavaderos de autos. Vale la pena tocar el tema acerca de aquellos, sólo para estar seguro.
 20. Probablemente notó que nuestra lista de los riesgos es de nivel alto. De nuevo, no descomponemos (dividimos) los riesgos a menos que podamos identificar los diferentes niveles de riesgos en los dos o más riesgos descompuestos que podemos nombrar. Si no podemos, entonces la descomposición de los riesgos significa el lanzamiento (prematadamente) en las pruebas de diseño. Sin embargo, cuando estamos trabajando desde un documento más detallado como una especificación de diseño, nosotros seríamos más específicos. En ese punto en el proyecto, ese nivel de especificación sería apropiado.

¿Cuáles defectos encontró en los requisitos? Es un buen beneficio secundario del análisis de los riesgos de calidad que sirve como una forma estructurada de revisión de los requisitos.

Ejercicio 2

Basado en su análisis de los riesgos de calidad para Omninet, esquematice un conjunto de juegos de prueba para abordar las áreas importantes de los riesgos.

Para cada juego de prueba, haga una lista breve.

- Los objetivos de las pruebas.
- Como reconocería las pruebas pasadas o falladas.

Establezca la relación entre sus juegos de prueba y los riesgos que serán cubiertos.

Basado en su análisis de los riesgos, ¿Cómo secuenciaría los juegos de prueba? ¿Qué otras consideraciones afectarían la secuenciación de los juegos de prueba?

Argumente.

Solución del Ejercicio 2

Juego de Prueba	Objetivos de las Pruebas y Criterios Paso/Falla	Cobertura de los Riesgos	Grupo de Secuencia
Funcionalidad del quiosco.	Encontrar defectos de funcionalidad en los quioscos individuales. Deberían proporcionar toda la funcionalidad de navegación disponible en una computadora personal con la misma velocidad de conexión.	1.000	1
Funcionalidad del centro de llamadas.	Encontrar defectos de funcionalidad en los quioscos individuales. Deberían proporcionar las funciones especificadas en el MRD – Documento de los Requisitos de Marketing – y SysRD – Documento de Requisitos del Sistema –, y no permitir cualquier otra operación.	1.000	1
Funcionalidad del centro de datos.	Encontrar defectos en las funciones individuales del centro de datos. Debería proporcionar funciones especificadas en el SysRD – Documento de los Requisitos del Sistema.	1.000	1

Juego de Prueba	Objetivos de las Pruebas y Criterios Paso/Falla	Cobertura de los Riesgos	Grupo de Secuencia
Funcionalidad de punta a punta.	Encontrar defectos en las operaciones que se originan en el quiosco, ir a través del centro de datos y llegar al centro de llamadas, y viceversa. Debería proporcionar las funciones especificadas en el MRD – Documento de los Requisitos de Marketing – y el SysRD – Documento de Requisitos del Sistema – y no permitir cualquier otra operación.	1.000 6.000	2
Seguridad y privacidad.	Encontrar defectos de seguridad, privacidad y confidencialidad. El quiosco, el centro de llamadas y el centro de datos no deberían permitir la programación o la penetración que divulguen datos confidenciales.	1.000	2
Fiabilidad.	Encontrar defectos de fiabilidad y disponibilidad del sistema. El quiosco, el centro de llamadas y el centro de datos deberían haber especificado la fiabilidad.	4.000	2
Sostenibilidad (“Supportability”).	Encontrar defectos en la capacidad de la actualización del sistema. Todas las actualizaciones especificadas deberían funcionar fiablemente y correctamente.	6.000	2

Juego de Prueba	Objetivos de las Pruebas y Criterios Paso/Falla	Cobertura de los Riesgos	Grupo de Secuencia
Localización	Encontrar defectos en idiomas adicionales y otros países. Todos los idiomas y las monedas compatibles deberían trabajar correctamente.	2.000	3
Rendimiento	Encontrar defectos en los tiempos de respuesta y la utilización de los recursos. El quiosco, el centro de llamadas y el centro de datos deberían responder a las especificaciones en todas las condiciones de carga.	5.000	3
Control de Errores	Encontrar defectos en la inhabilidad del sistema para controlar eventos previsibles del usuario, el sistema y el entorno. Los comportamientos del quiosco, el centro de llamadas y el centro de datos deberían degradarse elegantemente en condiciones de error.	1.000 4.000 6.000	3
Exploratorio	Encontrar defectos en las áreas que de otra manera no serían cubiertas por las pruebas. El quiosco, el centro de llamadas y el centro de datos deberían exhibir un comportamiento que los probadores encuentren razonable en todas las circunstancias.	1.000 2.000 3.000 4.000 5.000 6.000	3

Juego de Prueba	Objetivos de las Pruebas y Criterios Paso/Falla	Cobertura de los Riesgos	Grupo de Secuencia
Pruebas Beta	Encontrar defectos de funcionalidad, rendimiento y usabilidad. Las interacciones del quiosco y el centro de llamadas deberían satisfacer un subconjunto representativo de clientes potenciales.	1.000 2.000 3.000 4.000 5.000	Paralelo con la Prueba de Sistema

Tabla 4.3: Solución-Juegos de Pruebas

Note que hemos agrupado los juegos de prueba en grupos de secuencia que serían ejecutados en paralelo y en orden por los probadores. Si tuviera que haber orden dentro de un grupo, éste emergería de las restricciones logísticas como la disponibilidad de los probadores o del material de trabajo. Las restricciones logísticas podrían también resultar en un juego de prueba siendo ejecutado en paralelo con las pruebas en otro grupo de secuencia.

Las pruebas no están perfectamente en el orden de los riesgos, porque decidimos probar la funcionalidad individual, antes de empezar con pruebas más complejas. Idealmente, estas pruebas serían empujadas a las etapas tempranas de pruebas y repetidas sólo como criterios de entrada para la prueba de sistema.

4.2 Categorías de las Técnicas de Diseño de Pruebas

Objetivos del Aprendizaje

LO-4.2.1 Recordar las razones que tanto el método de pruebas basado en la especificación (caja negra) como en la estructura (caja blanca) son útiles para el diseño de casos de pruebas, y haga una lista de las técnicas comunes para cada uno. (K1)

LO-4.2.2 Explicar las características, cosas en común, y diferencias entre las pruebas basadas en la especificación, las pruebas basadas en la estructura y las pruebas basadas en la experiencia. (K2)

Esta sección, Categorías de las Técnicas de Diseño de Pruebas, cubrirá los siguientes conceptos clave:

- Las razones para las pruebas basadas en la especificación, basadas en la estructura y basadas en la experiencia.
- Técnicas comunes de caja negra y caja blanca.
- Características y diferencias entre las pruebas basadas en las especificaciones, pruebas basadas en la estructura y las pruebas basadas en la experiencia.

Hay tres tipos de técnicas de diseño de pruebas incluidas en el programa de estudios básico.

Algunas pruebas pueden ser clasificadas como basadas en la especificación. Esto es cuando principalmente creamos las pruebas por medio del análisis de las bases de pruebas. Cuando éstas fallan, estas pruebas revelan típicamente los defectos en la manera en la cual el sistema se comporta. Cuando éstas pasan, estas pruebas construyen típicamente la confianza en el comportamiento del sistema.

Algunas pruebas pueden ser clasificadas como basadas en la estructura, también llamadas pruebas de caja blanca. Esto es cuando creamos principalmente las pruebas por medio del análisis de la estructura del componente o sistema. Cuando éstas fallan, estas pruebas revelan típicamente los defectos en la manera en la cual el sistema está construido.

Cuando éstas pasan, estas pruebas construyen típicamente la confianza en la construcción del sistema.

Algunas pruebas pueden ser clasificadas como basadas en la experiencia, así como las basadas en los ataques, las basadas en las listas de comprobación y las exploratorias.

Glosario del ISTQB

Ataque: Intento dirigido y enfocado para evaluar la calidad, especialmente la fiabilidad, de un objeto de prueba intentando de forzar que ocurran fallas específicas. Véase también pruebas negativas. Note que este término no ha sido enunciado específicamente en esta sección pero es incluido aquí ya que es un sinónimo para ataque de defectos.

Ataque de defectos: Véase ataque.

Pruebas negativas: Pruebas con el objetivo de mostrar que una componente o sistema no funciona. Las pruebas negativas están relacionadas con la actitud de los probadores más que un método de pruebas específico o una técnica de diseño de pruebas, p.ej. las pruebas con valores de entrada inválidos o excepciones. Note que este término no es específicamente enunciado para esta sección pero es incluido aquí ya que está relacionado con el término ataque de defectos.

Esto es cuando creamos las pruebas principalmente basadas en la comprensión del sistema, la experiencia pasada y las suposiciones con cierta base acerca de los defectos. Cuando fallan, estas pruebas revelan típicamente los defectos en los lugares en los que otros sistemas tienen defectos. Sin embargo, debido a la falta de documentación, estas pruebas a menudo no construyen la confianza en el sistema, porque la cobertura es difícil de medir.

Hay un poco de ambigüedad acerca de estas categorías. El Programa de Estudios Nivel Básico dice que tanto las pruebas basadas en la especificación como las basadas en la experiencia son pruebas de caja negra o de comportamiento, las cuales pueden ser funcionales y no funcionales.

Sin embargo, muchos profesionales consideran a las pruebas basadas en la experiencia como toda una categoría separada, distinta a la clasificación de estructura/comportamiento. Esto es consistente con el Programa de Estudios Avanzado.

Las pruebas basadas en la especificación presentan algunos elementos básicos comunes. En primer lugar, utilizaremos modelos formales o informales para especificar el problema que debe ser resuelto, el software o sus componentes. A continuación, derivaríamos sistemáticamente los casos de prueba de estos modelos.

Ejemplos típicos de las pruebas basadas en la especificación incluyen:

- Partición de equivalencias y análisis de valores límite.
- Diagramas de transición de estados.
- Tablas de decisión.

Las pruebas basadas en la estructura o pruebas de caja blanca, presentan algunos elementos básicos comunes. Primero, utilizaremos las estructuras del sistema para derivar los casos de prueba, por ejemplo el código y diseño. Estas estructuras incluyen el código mismo, las tablas de base de datos, las consultas o las relaciones y el diseño del sistema. El siguiente paso típico sería que usted mida el grado de cobertura estructural para otros casos de pruebas existentes, así como las pruebas basadas en la especificación o las pruebas basadas en la experiencia. A continuación, si se justifica, utilizaremos más casos de prueba que pueden derivarse de forma sistemática para aumentar la cobertura.

Los ejemplos típicos de pruebas basadas en la estructura incluyen:

- Cobertura de sentencias.
- Cobertura de ramas o decisión.

Glosario del ISTQB

Cobertura de sentencia: El porcentaje de sentencias ejecutables que hayan sido ejercidas por un juego de pruebas.

Pruebas basadas en la estructura: Véase pruebas de caja blanca.

Las pruebas basadas en la experiencia presentan también algunos elementos básicos comunes. Primero, usted utiliza su conocimiento y experiencia utilizada para derivar casos de prueba. Puede utilizar el conocimiento y la experiencia del software, su utilización y su entorno o conocimiento acerca de los defectos históricos y probables y su distribución.

Ejemplos típicos de pruebas basadas en la experiencia incluyen:

- Ataques.
- Listas de comprobación.
- Exploratorias.

4.2.1 Ejercicios

Ejercicio 1

Refiérase a su esquema de juegos de prueba para Omninet.

Para cada juego de prueba, identifique si una, dos o todas de las siguientes tres categorías de las técnicas de pruebas serían útiles en el diseño de los casos de prueba:

- Basadas en la especificación (caja negra).
- Basadas en la estructura (caja blanca).
- Basadas en la experiencia.

Argumente.

Solución del Ejercicio 1

Juego de Prueba	Basada en la Especificación	Basada en la Estructura	Basada en la Experiencia
Funcionalidad del quiosco	Las técnicas principales de diseño de pruebas estarían basadas en la especificación.	Es útil comprobar la cobertura de las pantallas y los elementos de las pantallas.	Los probadores deberían utilizar su juicio mientras están ejecutando las pruebas.
Funcionalidad del centro de llamadas	Las técnicas principales de diseño de pruebas estarían basadas en la especificación.	Es útil comprobar la cobertura de la integración para los paquetes de software comercial de distribución masiva.	Los probadores deberían utilizar su juicio mientras están ejecutando las pruebas.
Funcionalidad del centro de datos	Las técnicas principales de diseño de pruebas estarían basadas en la especificación.	Utilice la cobertura del código para asegurar las pruebas rigurosas.	Los probadores deberían utilizar su juicio mientras están ejecutando las pruebas.

Juego de Prueba	Basada en la Especificación	Basada en la Estructura	Basada en la Experiencia
Funcionalidad de punta a punta	Las técnicas principales de diseño de pruebas estarían basadas en la especificación.	Utilice la arquitectura del sistema y la red para comprobar la cobertura de los caminos.	Los probadores deberían utilizar su juicio mientras están ejecutando las pruebas y también deberían variar las pruebas para cubrir nuevos escenarios.
Seguridad y privacidad	Todas las tres técnicas son igualmente útiles para estas clases de pruebas y deben ser aplicadas para asegurar la rigurosidad.		
Fiabilidad	Las especificaciones determinan los criterios de paso/falla.	La estructura determina la manera de crear carga.	La experiencia es necesaria para diseñar las pruebas de fiabilidad.
Sostenibilidad ("Supportability")	Las especificaciones determinan los criterios de paso/falla.	La estructura determina la manera de realizar las actualizaciones.	La experiencia es necesaria para diseñar las pruebas de compatibilidad.
Localización	Las especificaciones nos indican cuáles idiomas, monedas y zonas horarias deben ser probadas.	Descargar todos los mensajes y comprobar la traducción.	El juicio humano es necesario para interpretar que la localización de la Interfaz del Usuario ("UI") sea correcta.

Juego de Prueba	Basada en la Especificación	Basada en la Estructura	Basada en la Experiencia
Rendimiento	Las especificaciones determinan los criterios de paso/falla.	La estructura determina la manera de crear carga.	La experiencia es necesaria para diseñar las pruebas de rendimiento.
Control de errores	Útil, pero no todas las condiciones de error son especificadas. Algunas técnicas permiten el descubrimiento de errores no especificados.	La comprensión de la estructura y todos los posibles mensajes de error son clave para forzar las fallas.	El juicio humano es necesario para interpretar que el control de los errores sea correcto.
Exploratorio	Aplique las técnicas basadas en la especificación en tiempo real.	Aplique las técnicas basadas en la estructura en tiempo real.	Solo los probadores experimentados pueden realizar las pruebas exploratorias efectivamente.
Pruebas beta	Las pruebas no son prediseñadas, sin embargo pueden seguir los casos de uso.		

Tabla 4.4: Solución-Juegos de Pruebas en 3 categorías

Ejercicio 2

Usted está probando un sistema de comercio electrónico que vende chucherías como gorras, camisetas de baseball etc. El ejercicio consiste en crear pruebas funcionales para la página Web que acepta los pedidos. Una pantalla prototipo de la página Web para las entradas de los pedidos es mostrada en la figura 4.7.24

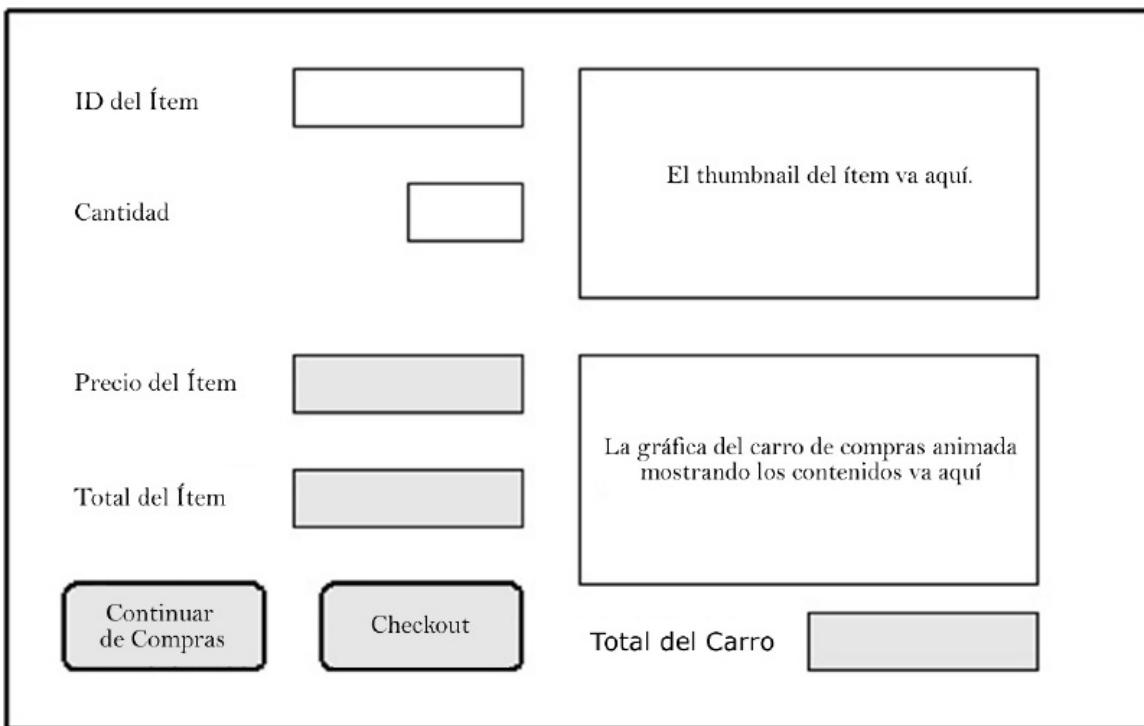


Figura 4.7: Plantilla de la página Web para las entradas de los pedidos del comercio electrónico de Omninet.

El sistema acepta como Identificador de un ítem a un valor numérico de 5 dígitos que va desde 00000 a 99999.

Estos identificadores de ítems están ordenados por el precio en el catálogo de los productos en la base de datos del sistema, donde los ítems más económicos tienen los números de Identificador de ítem más bajos (lo más cercano a 00000) y los ítems más costosos tienen los números de Identificador más altos (lo más cercano a 99999). Sin embargo, no se tiene que preocupar acerca de la realización de las pruebas del orden de los datos en la base de datos, porque no está probando el proceso de la entrada de los datos para el catálogo.

El sistema acepta una cantidad que debe ser pedida, del 1 al 99. Si el usuario ingresa un Identificador de ítem previamente pedido y una cantidad 0, ese ítem es retirado del carrito de compras.

Basado en estas entradas, el sistema recupera el precio del ítem, calcula el total del ítem (la cantidad de veces el precio), y adiciona el total del ítem al total del carrito. Debido a los límites de los pedidos con tarjetas de crédito, el máximo total para el carrito es de US\$ 999,99.

Su trabajo como probador es de utilizar el análisis de los valores límite y el particionamiento de las clases de equivalencia para crear las pruebas.

Solución del Ejercicio 2

Nosotros dibujamos las clases de equivalencia y los valores límite para el número del Identificador de ítem en dos maneras, como es mostrado en la figura 4.8. Hay dos maneras de pensar acerca de este campo. Una manera es que este campo sea un entero sin signo con un valor mínimo de 0 y un valor máximo de 99999. En ese caso, la representación lineal de los números en la parte superior de la figura tiene sentido. Tendrá que probar cuatro valores del Identificador de ítem, "-1", "0", "99999" y "100000".

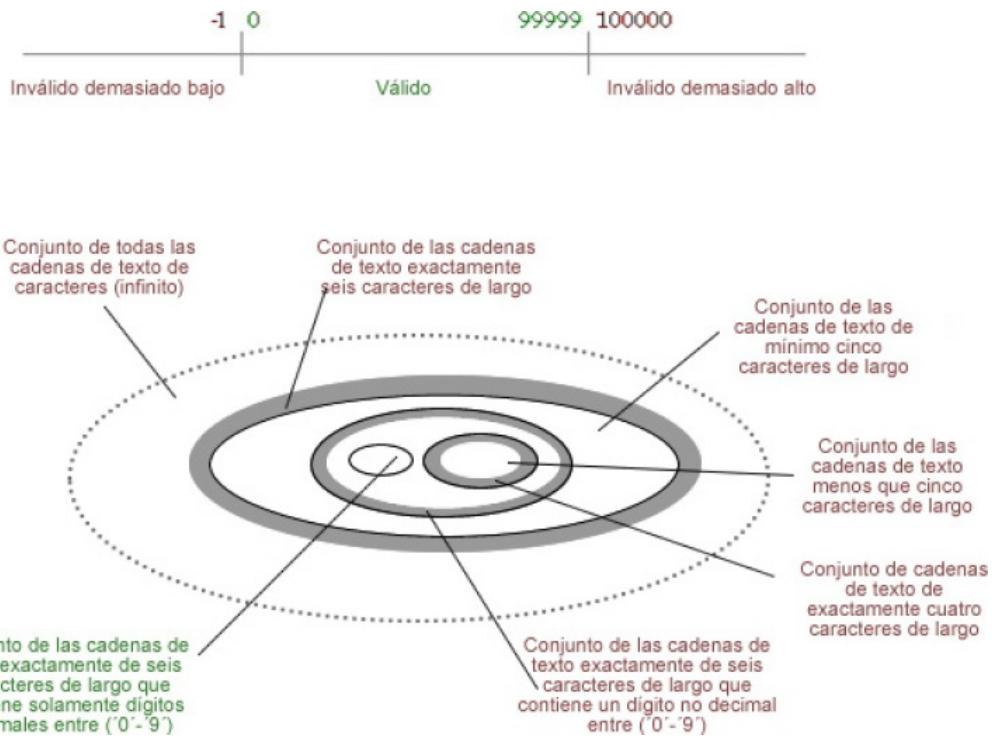


Figura 4.8: Clases de equivalencia y valores límite para el Identificador del ítem

La otra manera de pensar acerca de este campo es que éste es como una cadena de caracteres que deben ser exactamente 5 caracteres de largo y consistir solamente de dígitos decimales. En ese caso, la representación gráfica colocada abajo en la figura tiene sentido. Querrá probar una cadena realmente, realmente larga (para hacer un desborde de búfer), una cadena de 6 caracteres como "100000", (inválida solo por el largo), dos cadenas de cinco caracteres como "31:75" y "27/86" (inválida porque no es una cadena que consta solamente de dígitos decimales), una cadena de 4 caracteres como "9999" (inválida sólo por el largo), una cadena de cero caracteres como una cadena nula: "" (usualmente una entrada de carácter interesante), y dos cadenas de cinco caracteres como "00000" y "99999".

¿Cuál representación es correcta? Bien, si usted puede preguntar al programador cómo está implementado el campo, usted podría probarlo según lo que él le diga. Sin embargo, no lo hace más difícil asumir que ambos podrían estar correctos y probarlo de ambas maneras. En ese caso, si la implementación cambia, sus pruebas cubrirán todos los casos interesantes.

Dibujamos las clases de equivalencia y los valores límite para las cantidades pedidas como es mostrado en la figura 4.9. Observe cómo el valor cero es a veces válido y a veces inválido, dependiendo de que si el Identificador del ítem ha sido previamente ingresado.



Figura 4.9: Clases de equivalencia y valores límite para las cantidades pedidas.

Las entradas, las acciones y los resultados esperados de las pruebas son mostrados en las tablas de abajo.

Número de la Prueba	1	2	3	4	5
---------------------	---	---	---	---	---

Entradas, Acciones

Identificador del ítem	-1	0	00000	31:75	99999
Cantidad	1	1	99	2	1
Siguiente Acción	Continuar	Continuar	Continuar	Continuar	Pasar a pagar

Resultados Esperados

¿Mensaje de Error?	Si	No	No	Si	No
Precio del ítem (IP)	En blanco	Confirmar	Confirmar	En blanco	Confirmar
Total del ítem	En blanco	1*IP	99*IP	En blanco	1*IP
Contenidos del Carrito	Vacio	00000×1	00000×100	00000×100	00000×100 99999×1
Total del carrito	0,00	1*IP	+99*IP	=	+1*IP
Pasar a pagar	No	No	No	No	Si

Número de la Prueba	6	7	8	9	10
---------------------	---	---	---	---	----

Entradas, Acciones

Identificador del ítem	55555	27/86	12785	12785	63041
Cantidad	9	3	16	0	1
Siguiente Acción	Pasar a pagar	Pasar a pagar	Continuar	Pasar a pagar	Continuar

Resultados Esperados

¿Mensaje de Error?	No	Si (2)	No	Si	No
Precio del ítem(IP)	Confirmar	En blanco	Confirmar	En blanco	Confirmar
Total del ítem	9*IP	En blanco	16*IP	En blanco	1*IP
Contenidos del Carrito	55555×9	Vacio	12785×16	Vacio	63041×1
Total del carrito	9*IP	0,00	16*IP	0,00	1*IP
Pasar a pagar	Si	No	No	No	No

Número de la Prueba	11	12	13	14	15
---------------------	----	----	----	----	----

Entradas, Acciones

Identificador del ítem	70152	63041	70152	100000	Realmente largo
Cantidad	5	0	2	1	17
Siguiente Acción	Continuar	Continuar	Pasar a pagar	Continuar	Continuar

Resultados Esperados

¿Mensaje de Error?	No	No	No	Si	Si
Precio del ítem (IP)	Confirmar	En blanco	Confirmar	En blanco	En blanco
Total del ítem	5*IP	En blanco	2*IP	En blanco	En blanco
Contenidos del Carrito	63041×1 70152×5	70152×5	70152×7	Vacio	Vacio
Total del carrito	+5*IP	-1*IP ₆₃₀₄₁	+2*IP	0,00	0,00
Pasar a pagar	No	No	Si	No	No

Número de la Prueba	16	17	18	19
Entradas, Acciones				
Identificador del ítem	9999	Nulo	Llenar el carrito hasta el límite	Llenar el carrito pasado el límite con 0.01
Cantidad	3	5		
Siguiente Acción	Continuar	Continuar	Pasar a pagar	Pasar a pagar
Resultados Esperados				
¿Mensaje de Error?	No	Si	No	No
Precio del ítem (IP)	Confirmar	En blanco	Confirmar	Confirmar
Total del ítem	3*IP	En blanco	Confirmar	Confirmar
Contenidos del Carrito	3x09999	Vacio	Confirmar que está al tope	Confirmar (casi) al tope
Total del carrito	3*IP	0,00	999,99	(<)= 999,99
¿Pasar a pagar?	No	No	No	Si

Número de la Prueba	20	21	22
Entradas, Acciones			

Identificador del ítem	34572	23897	89457
Cantidad	0	-1	100
Siguiente Acción	Continuar	Continuar	Continuar

Resultados Esperados			
¿Mensaje de Error?	Si	Si	Si
Precio del ítem (IP)	En blanco	En blanco	En blanco
Total del ítem	En blanco	En blanco	En blanco
Contenidos del Carrito	Vacio	Vacio	En blanco
Total del carrito	0.00	0.00	0.00
¿Pasar a pagar?	No	No	No

Tabla 4.5: Solución-Casos de Prueba

Hemos asumido que el sistema aceptará el Identificador del ítem ya sea como entero (sin ceros por delante) o como un campo de cinco caracteres con caracteres numéricos solamente (con ceros por delante). Esa es la manera permisiva de definir el sistema, y pensamos que los sistemas deberían ser permisivos cuando sea posible.

Hemos instruido a los probadores que confirmen los precios, probablemente utilizando un catálogo en papel o una lista de precios y una calculadora. Cuando un oráculo fiable de pruebas sea fácilmente de obtener y cambiante en el tiempo, es mejor no introducir redundancia y discrepancias en última instancia por medio de la codificación en duro del resultado esperado en el caso de prueba. En cambio, puede hacer una nota para comprobar el resultado esperado durante la ejecución de las pruebas y posiblemente proporcionar alguna información acerca de dónde encontrar el oráculo de pruebas.

En la prueba 3, podríamos haber notado que estamos asumiendo que usted puede pedir 100 (o más) de un solo ítem simplemente ingresando dos pedidos. ¿Debería estar bien eso? Bien, si el objetivo del límite es prevenir que la gente pida accidentalmente demasiados de un solo ítem, forzando al usuario a ingresar dos pedidos hace obvio que la gran cantidad de pedidos es intencionada. Si el sistema rechazó este valor con un mensaje como, "Lo siento, pero usted sólo puede pedir una cantidad total de 99 de cualquier ítem dado", entonces lo más probable es que no sería un error. Sin embargo, necesitaría de adicionar otra prueba donde usted pediría la cantidad de 99 de un ítem en un sólo pedido.

También en la prueba 3, el "+" por delante en el total del carrito indica que usted debería esperar el total del carrito para incluir el total previo del carrito mas la adición del total del ítem para este ítem. En la prueba 4, el "=" por delante en el total del carrito indica que usted debería esperar que el total del carrito no sea modificado del total anterior del carrito.

En la prueba 5, estamos asumiendo que el total del ítem para este ítem—el más costoso en el catálogo— junto con los 100 ítems menos costosos, no excederá el límite total del pedido de US\$ 999,99. Si excede, entonces nosotros posiblemente tendríamos que eliminar ítems del carro o pasar a pagar antes de ejecutar la prueba 5.

En la prueba 6—y en verdad, en todas las pruebas— estamos asumiendo que el Identificador del ítem que hemos ingresado corresponde a un ítem real en el catálogo. Es posible que los Identificadores de ítem pudieran ser escasos, en el sentido de que la mayoría de los números válidos de Identificación de ítem no corresponden realmente a un ítem en el catálogo. Es ese caso, la prueba tendría que elegir el Identificador del ítem más cercano posible para alcanzar el mismo efecto para muchas de las pruebas. Usted también quisiera comprobar lo que ocurre cuando usted ingresa un Identificador de ítem que es válido pero no está disponible en el catálogo.

Note que hemos ejercido los valores límite sobre las acciones y las salidas, no sólo entradas. Por ejemplo, la prueba 6 verifica el pago con un sólo ítem en el carro. Las pruebas 7 y 9 verifican el intento de pago sin ítems en el carro. La prueba 7 debería resultar en dos mensajes de error, uno por un Identificador incorrecto del ítem y uno por el intento de pago con el carro vacío. La prueba 9 debería resultar en un mensaje de error individual por pagar con el carro vacío. Otras pruebas verifican el pago con múltiples ítems en el carro.

En la prueba 13, hemos asumido que ingresando un Identificador del ítem y una cantidad válida para algo que ya está en el carro de compras, se adiciona al total para ese ítem. Sin embargo, éste podría sobrescribir el total. Sin una clara explicación de por qué el sistema debería sobrescribir en vez de añadir al total, nosotros informaríamos eso como un defecto. Como un usuario, esperaríamos que se adicione, no se sobrescriba, así que la decisión estará reflejada en nuestra interpretación del resultado de la prueba.

No existe un límite establecido acerca de cuántos ítems pueden estar en el carro, pero existe un límite establecido acerca el costo total de los ítems. Eso está cubierto por las pruebas 18 y 19. En la prueba 18, el sistema debería permitirnos comprar US\$ 999,99 en valor total de ítems. En la prueba 19, el sistema debería rechazar el último ítem que adicionamos—ése que pone el total de nuestro carro tan cerca como sea posible a US\$ 1.000,00— pero debería permitirnos comprar los otros ítems que ya hemos añadido a nuestro carro.

Hasta cierto punto, este ejemplo deja al criterio del probador o la especificación de los requisitos las preguntas de que si sería correcto un punto decimal al estilo europeo o un símbolo de moneda en el resultado. Usualmente preferimos tener probadores inteligentes que tomen estas decisiones a medida que ellos ejecutan las pruebas. De esa manera, si después el sistema se ubica en otro local, entonces no necesitamos cambiar las pruebas.

Ejercicio 3

Refiérase a la tabla de decisión de los pagos en el Documento de los Requisitos del Sistema Omninet.²⁵

Desarrolle escenarios de pruebas los cuales cubran adecuadamente la tabla de decisión mostrada.

¿Cómo determinó el nivel de cobertura necesaria? ¿Cuáles suposiciones de la implementación podrían cambiar el número de las pruebas necesarias?

Argumente.

Solución del Ejercicio 3

Cuando está comenzando a diseñar las pruebas desde una tabla de decisión, revise la técnica. La regla usual para la cobertura de una tabla de decisión es, “Por lo menos una prueba por columna”. Si las condiciones o acciones tienen valores límite, es posible que usted desee cubrir aquellas, las cuales podrían resultar en dos reglas por columna. Si las reglas pueden interactuar, entonces usted debería analizar y probar la interacción también, después de probar cada regla por sí misma.

También necesita considerar el análisis de los riesgos. En un ejercicio del análisis de los riesgos de calidad, identificamos el “pago válido rechazado/pago inválido aceptado” como un riesgo técnico muy bajo pero como un riesgo de negocios muy alto. Basado en eso, decidimos que este ítem de riesgo debería recibir pruebas amplias.

¿Se recordó usted de referirse a su análisis de los riesgos de calidad para guiarse acerca de cómo debería probar rigurosamente esta función? Si está realizando pruebas basadas en los riesgos, eso es importante. Usted fácilmente puede desalinear sus pruebas con el nivel de riesgo durante el diseño y el desarrollo de las pruebas. En esos casos, probará en exceso algunos ítems de riesgo bajo y probará demasiado poco algunos ítems de alto riesgo. En el capítulo posterior acerca de la trazabilidad, examinaremos una manera de comprobar para asegurar que sus pruebas permanecen alineadas con su evaluación de los riesgos durante el diseño y el desarrollo de las pruebas. En las pruebas basadas en los riesgos, su análisis de los riesgos de calidad es su guía básica.

Por supuesto, su evaluación de los riesgos puede cambiar durante el proyecto. Si, durante el diseño, el desarrollo o la ejecución de pruebas, usted obtiene nueva información o percepciones que le dicen que la evaluación de los riesgos está incorrecta, usted debería revisar la evaluación de los riesgos en vez de seguir una guía básica incorrecta.

Condición	Reglas de Negocios					
	1	2	3	4	5	6
Dinero válido	No	Si	-	-	-	-
Tarjeta válida de crédito/débito	-	-	No	Si	Si	Si
PIN válido (para débito) o PIN no requerido (para crédito)	-	-	-	No	Si	Si
Cantidad aprobada	-	-	-	-	No	Si
<i>Acción</i>						
Rechazar efectivo	Si	No	No	No	No	No
Rechazar tarjeta	No	No	Si	Si	No	No
Pedir una cantidad menor	No	No	No	No	Si	No
Vender período de tiempo	No	Si	No	No	No	Si

Tabla 4.6: Solución-Proceso de Pago de Omninet.

Las pruebas amplias para esta tabla de decisión, significan que cada condición en la tabla de decisión sea cubierta (véase la tabla 4.7). Las reglas 5 y 6 involucran una condición compuesta, "PIN válido (para débito) o PIN no requerido (para tarjeta de crédito)". Entonces necesitamos una prueba para cada una de las reglas desde la regla 1 a la 4, mientras que las reglas 5 y 6 necesitan dos pruebas cada una.

#	Acción y Datos del Probador	Resultado Esperado
1	Insertar dinero falso (p.ej. cortar un pedazo de papel en la forma y tamaño de un billete de US\$ 20, pintarlo de verde e insertarlo en la ranura para el dinero).	El dinero falso es rechazado. El quiosco no le da al usuario ningún tiempo de navegación.
2	Insertar dinero válido (p.ej. un billete de US\$ 20).	El dinero es aceptado. El quiosco le da al usuario el período de tiempo correspondiente. (Para determinar cuántos minutos son, consultar en la hoja de tasas).
3	Insertar una tarjeta inválida (p.ej. una tarjeta de viajero frecuente o una licencia de conducir).	La tarjeta es rechazada. El quiosco informa al usuario que la tarjeta no es una tarjeta válida de crédito o débito. El quiosco no le da al usuario ningún tiempo de navegación.
4	1. Insertar una tarjeta de débito válida. 2. Ingresar un PIN inválido. 3. Ingresar una cantidad válida (Es decir, un período válido de tiempo).z	La tarjeta es rechazada. El quiosco le informa al usuario que la red no validó la tarjeta. El quiosco no le da al usuario ningún tiempo de navegación.
5	1. Insertar una tarjeta de débito válida. 2. Ingresar un PIN válido. 3. Ingresar una cantidad que no sea un período válido de tiempo.	La tarjeta es aceptada. El quiosco le informa al usuario que el período de tiempo pedido no está permitido. El quiosco pide una cantidad válida de período de tiempo. El quiosco no le da al usuario ningún tiempo de navegación.
6	1. Insertar una tarjeta de crédito válida. 2. Ingresar una cantidad válida de período de tiempo que excede el límite del saldo disponible.	La tarjeta es aceptada. El quiosco informa al usuario que la cantidad pedida excede el saldo disponible. El quiosco pide una cantidad menor. El quiosco no da al usuario ningún tiempo de navegación.
7	1. Insertar una tarjeta de débito válida. 2. Ingresar un PIN válido. 3. Ingresar una cantidad que sea tanto una cantidad de período permitido como dentro del saldo disponible de la cuenta.	La tarjeta es aceptada. El quiosco da al usuario el período de tiempo apropiado. (Para determinar cuántos minutos son, consultar en la hoja de tasas).
8	1. Insertar una tarjeta de crédito válida. 2. Ingresar una cantidad que sea tanto una cantidad de período permitido como dentro del saldo disponible de la cuenta.	La tarjeta es aceptada. El quiosco da al usuario el período de tiempo apropiado. (Para determinar cuántos minutos son, consultar en la hoja de tasas).

Tabla 4.7: Solución-Pruebas del Proceso de Pagos de Omninet

Durante el diseño de estas pruebas, encontramos un defecto potencial en la tabla de decisión. La condición “Cantidad aprobada,” puede ser realmente falsa por dos razones. Primero, la cantidad pudo exceder la cantidad disponible en la cuenta, lo cual es lo que la tabla prevé.

Sin embargo, lo que aparentemente falta es la posibilidad de que el usuario pudiese introducir una cantidad que no corresponda a un período de tiempo permitido. Por ejemplo, si el tiempo es vendido en incrementos de 10 minutos a un dólar, entonces, ¿Qué debería hacer el quiosco si el usuario ingresa una cantidad como US\$ 1,75?

Entonces la acción “Pedir una cantidad menor”, no es suficiente. En realidad la acción debería ser, “Pedir una cantidad válida”.

Desde el punto de vista de las pruebas, quisiéramos probar en ambas maneras que la condición no podría ser cumplida en la regla 5. Sin embargo, por que no hay razón para pensar que el uso de una tarjeta de crédito o débito influenciaría en la lógica del quiosco en cuanto a períodos de tiempo válidos, no hay necesidad de incrementar el número de pruebas para intentar todas las permutaciones.

Desde el punto de vista del diseño del sistema, pudimos prevenir el problema utilizando una lista desplegable para pedir al usuario una cantidad. La lista solo contendría cantidades válidas.

Otra cuestión de prueba es que necesitaremos algunos accesorios para esta prueba. Necesitamos por lo menos una tarjeta de débito válida y una tarjeta de crédito válida. Estas tarjetas deben estar enlazadas a las cuentas que tengan los saldos apropiados. Obtener este tipo de accesorios de pruebas es a menudo difícil, entonces usted debería empezar a ver la forma cómo conseguirlos tan pronto como se entere que los necesitará.

Ejercicio 4

Los quioscos públicos de acceso al Internet de Omninet, pueden estar en varios estados, basados en la recepción de los pagos, las sesiones activas y así sucesivamente.²⁶

Refiérase al Documento de los Requisitos del Sistema y de Marketing de Omninet.

Dibuje un diagrama de estados para el quiosco.

Cubra el diagrama de estados con casos de prueba.

Argumente.

Solución del Ejercicio 4

Primero, dibujamos el diagrama de transición de estados, como es mostrado en la figura 4.10. Tomamos el inicio, la actualización, y la actualización de la cuenta, pero omitimos el filtrado. Algunas personas dibujan este diagrama con un estado de Comprobación del Contenido, ingresado por medio de un evento “enviar URL” durante la navegación, que sale y vuelve al estado de Navegación con los eventos “Aceptar URL” y “Rechazar URL”. De manera alternativa, algunas personas muestran dos pares evento/acción que se ejecutan en un bucle donde la URL [correcta] y URL [incorrecta] son los pares evento/condición que determinan qué bucle tomar. Adicionalmente, a otros les gusta permitir un evento de Expiración de Tiempo del estado Tratando la Expiración y Comprobando el Contenido.

Usted también pudo dibujar un diagrama más simple como algunos lo hacen, dejando de lado las acciones de inicialización así como también las de filtrado.

Observe que el modelo ve el mundo desde el punto de vista del quiosco. Las transiciones en este diagrama están definidas en el Documento de los Requisitos de Marketing y el Documento de los Requisitos del Sistema.

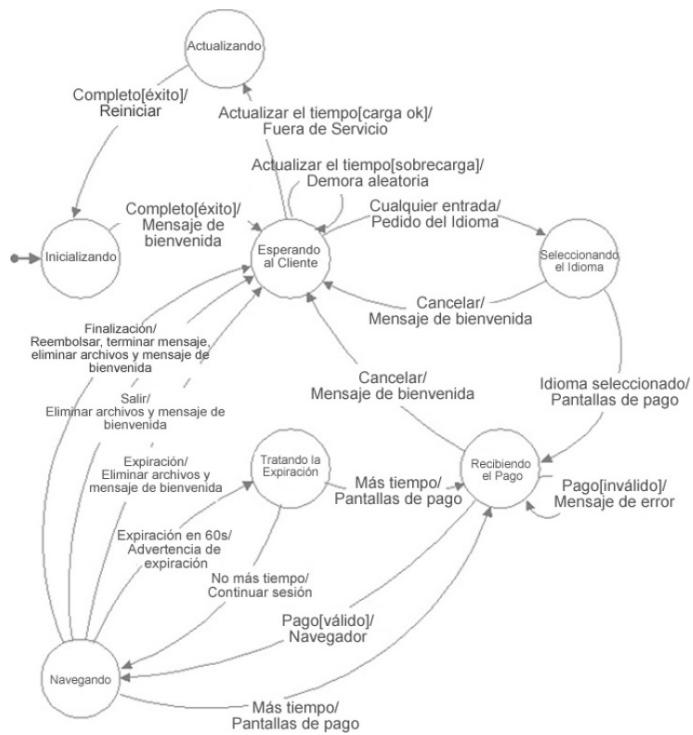


Figura 4.10: Diagrama de Transición de Estados del Quiosco

Durante la elaboración del diagrama, descubrimos las siguientes cuestiones en el Documento de los Requisitos de Marketing y el Documento de los Requisitos del Sistema:

1. ¿Qué ocurre si el sistema encuentra un problema serio mientras se está ejecutando? ¿Debería tratar de inicializarse por sí mismo? ¿Debería recibir el cliente—si alguno—un reembolso?
2. ¿Qué ocurre si el quiosco no se inicializa correctamente? Hubo un problema con una nave espacial de Marte donde la falla en el momento de inicializar, después de un problema serio (fuera del espacio del archivo) condujo a un sinfín de secuencias de reinicios.
3. ¿Qué pasa si una actualización falla?
4. Hemos asumido que el quiosco sólo puede empezar a buscar actualizaciones cuando esté en estado inactivo. Sin embargo, eso no está claro en los requisitos. Esto debería estar claro.

Asumimos que el informe de los estados por hora para el servidor debería ocurrir en segundo plano. También asumimos que el registro de los eventos de seguridad ocurre en segundo plano. Estos involucrarían diagramas de transición de estados diferentes. El diagrama mostrado, sólo aborda los eventos y las actividades en primer plano.

Elegimos representar las actividades del pago como estado único. También pudimos haber dividido esto en una secuencia de estados. Cuando está utilizando las técnicas de pruebas basadas en los estados, algunas veces tenemos que elegir dónde enfocar las pruebas y dónde simplificar.

Este diagrama de transición de estados cubre muchos de los riesgos identificados en el análisis de los riesgos. (En su mayor parte, éste cubre estos riesgos de manera indirecta). Algunos de aquellos riesgos requieren pruebas extensivas. Por eso, tiene sentido intentar cada combinación posible de los estados y eventos para este diagrama. En la situación donde las pruebas basadas en los estados se relacionan sólo con ítems de riesgo medio y bajo, entonces simplemente cubriendo el diagrama puede ser suficiente.

La diferencia básica entre cubrir el diagrama y cubrir la tabla es que la tabla toma en cuenta las situaciones que “no pueden ocurrir” y “no deberían ocurrir”. Donde se necesiten pruebas extensivas o control considerable de errores y pruebas de robustez, entonces usted debería planificar para cubrir la tabla.

Finalmente, hemos creado los escenarios de las pruebas para cubrir el diagrama de transición de estados. Recuerde que la regla de cobertura para el diagrama de transición de estados es visitar cada estado y atravesar cada transición. Una vez que recibimos la información acerca de cómo debieron ser tratadas las situaciones indefinidas en la tabla, nosotros añadiríamos éstas como escenarios adicionales.

#	Escenario	Resultado Esperado
1	El quiosco comienza. 1. Iniciar el sistema. 2. Tocar el teclado. 3. Seleccionar el idioma.	1. Comprobar la pantalla de Bienvenida. 2. Comprobar la indicación del Idioma. 3. Comprobar las pantallas de Pago.

	4. Hacer un pago válido. 5. Permitir al reloj 60 segundos de expiración. 6. No seleccionar más tiempo. 7. Permitir tiempo de expiración.	4. Comprobar el navegador. (Nota: El navegador debería ser seleccionado por el usuario.) 5. Comprobar la advertencia de Expiración. 6. Comprobar el retorno al Navegador, con la sesión ininterrumpida. 7. Comprobar la pantalla de Bienvenida. Verificar la limpieza de los archivos.
2	El quiosco comienza en la pantalla de Bienvenida. 1. Tocar el teclado. 2. Cancelar la transacción.	1. Comprobar la indicación del Idioma. 2. Comprobar la pantalla de Bienvenida.
3	El quiosco comienza en la pantalla de Bienvenida. 1. Repetir el escenario 2 pero no cancelar. En cambio, seleccionar el idioma. 2. Cancelar la transacción.	1. Comprobar las pantallas de Pago. 2. Comprobar la pantalla de Bienvenida.
4	El quiosco comienza en la pantalla de Bienvenida. 1. Repetir el escenario 3, pero no cancele. En cambio, realice un pago válido. 2. Realizar un pago válido. 3. Antes de que aparezca la advertencia de 60 segundos, compre más tiempo. 4. Ingrese un pago válido. 5. Antes de que aparezca la advertencia de 60 segundos, salga del sistema.	1. Comprobar el mensaje de Error. 2. Comprobar el Navegador. (Nota: El navegador debería ser seleccionado por el usuario.) 3. Comprobar las pantallas de Pago. 4. Comprobar el retorno al Navegador, con la sesión ininterrumpida. 5. Comprobar la pantalla de Bienvenida. Verificar la limpieza de los archivos.
5	El quiosco comienza en la pantalla de Bienvenida. 1. Repetir el 4, pero permitir que la advertencia de los 60 segundos aparezca y comprar más tiempo. 2. Antes de la advertencia de los 60 segundos, haga que un agente del centro de llamadas termine la sesión.	1. Comprobar el navegador. (Nota: El navegador debería ser seleccionado por el usuario.) Comprobar las pantallas de pago. 2. Comprobar el reembolso y el mensaje de terminación. Comprobar la pantalla de Bienvenida. Verificar la limpieza de los archivos.
6	El quiosco comienza en la pantalla de Bienvenida. Ajuste el tiempo local un poco antes de las 2:00 AM. No cargue el servidor de actualización. 1. Permitir que el reloj llegue a las 2:00 AM. 2. Permitir que la actualización se complete exitosamente. 3. Permitir que la inicialización se complete exitosamente.	1. Comprobar el comienzo de la actualización. Verificar el mensaje Fuera de servicio. 2. Verificar el reinicio. 3. Comprobar la pantalla de Bienvenida.
7	Repetir la configuración del escenario 6, pero cargue intensamente el servidor de actualizaciones. 1. Repetir el escenario 6. 2. Eliminar la carga intensa del servidor de actualizaciones. Dejar que la demora expire y que la actualización comience. 3. Permitir que la actualización se complete exitosamente. 4. Permitir que la inicialización se complete exitosamente.	1. Comprobar la demora aleatoria. 2. Verificar el mensaje de de Fuera de servicio. 3. Verificar el reinicio. 4. Comprobar el mensaje de Bienvenida.

Tabla 4.8: Solución-Pruebas de estado del quiosco

Usted podría, como un conjunto de pruebas adicional, ejercer bucles en el diagrama de estados. Por ejemplo, usted pudo continuar en el escenario 7 por algunas horas para mantener la carga en el servidor, verificando múltiples retrasos aleatorios.

Si puede automatizar las pruebas—que podría en este caso resultar algo desafiante—usted podría

tener una herramienta que continuamente navegue a través de los estados, creando ambas combinaciones de estados/eventos definidas e indefinidas, para ver cómo se comporta el sistema. Usted puede descubrir problemas de agotamiento de memoria, bloqueos y caídas intermitentes, y otros tales defectos con tales pruebas.

4.3 Técnicas Basadas en la Especificación

Objetivos del Aprendizaje

LO-4.3.1 Escribir casos de prueba de modelos de software dados utilizando particionamiento de equivalencia, análisis de valores límite, tablas de decisión y diagramas/tablas de transición de estados. (K3)

LO-4.3.2 Explicar el propósito principal de cada una de las cuatro técnicas, cuál nivel y tipo de pruebas podría utilizar la técnica y cómo la cobertura puede ser medida. (K2)

LO-4.3.3 Explicar el concepto de las pruebas de casos de uso y sus beneficios. (K2)

Glosario del ISTQB

Análisis de valor límite: Una técnica de diseño de pruebas de caja negra en la cual los casos de prueba son diseñados basados en los valores límite. Véase también valor límite.

Valor límite: Un valor de entrada o valor de salida el cual se encuentra en el borde de una partición de equivalencia o en la distancia incremental más pequeña en cada lado de un borde, por ejemplo, el valor mínimo o máximo de un rango. Tenga en cuenta que este término no se enunció específicamente para esta sección, pero se lo incluye aquí, ya que es esencial para comprender el término análisis de valor límite.

Pruebas de tabla de decisión: Una técnica de diseño de pruebas de caja negra en la cual los casos de prueba son diseñados para ejecutar las combinaciones de entradas y/o estímulos (causas) representadas en una tabla de decisión. Véase también tabla de decisión.

Tabla de decisión: Una tabla que muestra las combinaciones de entradas y/o estímulos (causas) con sus productos y/o acciones (efectos) asociados, los cuales pueden ser utilizados para diseñar casos de prueba. Tenga en cuenta que este término no se enunció específicamente para esta sección, pero se lo incluye aquí, ya que es esencial para comprender el término prueba de tablas de decisión.

Particionamiento de equivalencia: Una técnica de diseño pruebas de caja negra en la cual los casos de prueba son diseñados para ejecutar los representantes de las particiones de equivalencia. En principio, los casos de prueba son diseñados para cubrir cada partición por lo menos una vez.

Pruebas de transición de estados: Una técnica de diseño de pruebas de caja en la cual los casos de prueba son diseñados para ejecutar transiciones de estado válidas e inválidas. Véase también pruebas de comutador de multiplicidad.

Transición de estados: Una transición entre dos estados de un componente o sistema. Tenga en cuenta que este término no se enunció específicamente para esta sección, pero se lo incluye aquí, ya que es esencial para comprender el término pruebas de transición de estado.

Pruebas de casos de uso: Una técnica de diseño de pruebas de caja negra en la cual los casos de prueba son diseñados para ejecutar los escenarios de los casos de uso.

Caso de uso: Una secuencia de transacciones en un diálogo entre un actor y una componente o sistema con un resultado tangible, donde un actor puede ser un usuario o cualquier cosa que pueda intercambiar información con el sistema. Tenga en cuenta que este término no se enunció específicamente para esta sección, pero se lo incluye aquí, ya que es esencial para comprender el término pruebas de casos de uso.

Esta sección, Técnicas Basadas en la Especificación, cubrirá los siguientes conceptos clave:

- Cómo escribir casos de prueba a partir de los modelos de software dados, utilizando el particionamiento de equivalencias, el análisis de valores límite, las tablas de decisión y los diagramas de transición de estados.
- El principal objetivo de cada técnica y cómo la cobertura puede ser medida.
- Las ideas esenciales de las pruebas de casos de uso.

Una prueba basada en la especificación es aquella que está diseñada a partir de la especificación del sistema. Esa es una definición un tanto circular e inútil. Es más útil recordar que otros nombres comunes para las pruebas basadas en la especificación son las pruebas de caja negra o las pruebas de comportamiento. Intuitivamente, podemos pensar en una prueba de caja negra, como una donde ignoramos el funcionamiento interno del sistema y nos enfocamos en cómo se supone que debería comportarse. El comportamiento en cuestión puede ser funcional— ¿Qué hace?—O no funcional— ¿Cómo lo hace? —basado en la clasificación del ISO 9126.

La primera técnica basada en la especificación es el particionamiento de equivalencias. Esto es básicamente una manera elegante de describir algo que la mayoría de los probadores hacen todo el tiempo, aunque por lo general no del todo.

El primer paso en el particionamiento de equivalencias es la identificación de alguna entrada, salida, comportamiento o algún entorno que necesita probar. Luego usted divide el conjunto de todos los valores, comportamientos, configuraciones, opciones o lo que usted tiene como subconjuntos y espera que el sistema los maneje de manera equivalente.

Esta expectativa de equivalencia debería surgir ya sea de alguna especificación que requiere equivalencia, o de una ley o estándar que requiere equivalencia, o simplemente del sentido común o de expectativas razonables. Estos subconjuntos son llamados ya sea clases de equivalencia o particiones de equivalencia.

Típicamente, para cualquier conjunto dado de casos de prueba que necesita construir, necesita usualmente probar más de una entrada, salida, comportamiento, etc. Entonces, repetirá este

proceso de identificar y particionar²⁷ (“to partition”) conjuntos hasta que haya creado las particiones de equivalencia para todas las entradas, salidas, comportamientos, y así sucesivamente.

Ahora, en este punto, usted crea sus casos de prueba. Para construir las pruebas para las entradas, las salidas, las configuraciones o las opciones válidas o lo que sea con lo que esté trabajando, seleccione un valor de cada partición de equivalencia válida, a continuación defina el resultado esperado en esa situación. Repita este proceso hasta que haya seleccionado al menos un valor representativo para cada partición de equivalencia válida en todas las clases de equivalencia que ha creado.

Note que hemos dicho al menos una vez. Hay muchos casos donde podría seleccionar más de un valor representativo. Por ejemplo, en el caso de las clases definidas en conjuntos que están ordenados, podría seleccionar dos valores para cada clase, específicamente aquellos en el límite superior e inferior de cada clase. Estos valores se denominan valores límite, y hablaremos más acerca de esta técnica en un momento.

Además, podría seleccionar múltiples miembros de una partición cuando la partición es particularmente importante desde una perspectiva de negocios. Alternativamente, si considera que los defectos son particularmente probables para una cierta partición, podría seleccionar múltiples miembros de esa partición. Al diseñar las pruebas, remítase hacia atrás a su análisis de los riesgos para determinar el grado adecuado de cobertura de las pruebas.

Para construir las pruebas para las entradas, las salidas, las configuraciones o las opciones de ajuste inválidas o lo que sea con lo que esté trabajando, seleccione un valor de una—y sólo una—partición inválida de equivalencia. Para el resto de las particiones de equivalencia, seleccione un valor válido. La razón de esta regla es que, si prueba valores inválidos juntos, no podrá estar seguro de que una entrada, salida, comportamiento o lo que sea por alguna razón **específica** serán tratados correctamente. Por supuesto, si sospecha que ciertos valores inválidos van a interactuar, puede añadir pruebas para ello, pero asegúrese de que también ha probado individualmente cada valor inválido.

Después que ha seleccionado los valores válidos e inválidos, como antes, usted define el resultado esperado en esa situación. Usted repite este proceso hasta que se haya seleccionado al menos un valor representativo para cada partición de equivalencia inválida en todas las clases de equivalencia que ha creado.

¿Qué tal un ejemplo? Suponga que usted está probando impresoras compatibles para una aplicación.

Una forma de abordar el particionamiento de un conjunto grande—de hecho enorme—de impresoras en el mundo es crear particiones basadas en la interfaz física, es decir, cómo la impresora se conecta físicamente a la computadora. Algunas impresoras tienen interfaces paralelas, algunas seriales, algunas USB, algunas infrarrojas, y así sucesivamente. Tenga en cuenta que incluso podemos crear dos sub particiones de la partición USB: USB 1.1 y USB 2.0.

Otra forma de crear particiones del conjunto de impresoras se basa en la interfaz lógica, es decir, ¿Cómo son los datos enviados a la impresora, empaquetados de manera compacta, en algún flujo importante? Algunas impresoras utilizan PostScript para recibir datos, algunos HPPL, algunos ASCII, y algunos utilizan otros formatos.

Y aún otra manera de particionar el conjunto de impresoras se basa en la aplicación de la imagen, es decir, ¿Cómo es colocada la tinta en el papel? Algunas impresoras utilizan chorros láser, otras chorros de tinta, otras chorros de burbujas, otras matriz de puntos, otras son impresoras de línea, otras utilizan bolas giratorias de letras, otras utilizan bolígrafos, y así sucesivamente.

Tenga en cuenta que podemos escoger la forma en la cual quisiéramos particionar basados en lo que estamos tratando de encontrar, ya sea los defectos o construir la confianza o reducir el riesgo. Si se asegura que la imagen se ve exactamente como la pantalla es esencial, entonces la aplicación de la imagen es el particionamiento esencial. Si se asegura que los drivers de las impresoras están funcionando correctamente, entonces la interfaz lógica es la partición esencial. O, si se asegura que el hardware es compatible, entonces la interfaz física es el particionamiento esencial. Por supuesto, podría dividir las tres formas, y asegurarse de que tiene al menos una impresora para cada partición de equivalencia identificada.

Volvamos a los valores límite. El análisis de valores límite es un refinamiento del particionamiento de equivalencias que selecciona los bordes o puntos finales de cada partición para las pruebas.

Hay algo bueno acerca de las pruebas de valores límite, cuando pueda hacerlo. Observe que el particionamiento de equivalencia puede encontrar defectos en el código que maneja cada partición de equivalencia. En otras palabras, ¿Se comporta el programa de la manera Y cuando debería comportarse de la manera X? Sin embargo, un valor arbitrario de la partición de equivalencia no comprueba necesariamente para ver que el **programa** reconoce correctamente los miembros de la partición de equivalencia. Los valores límite, porque comprueban los límites y son miembros de las clases de equivalencia, pueden también encontrar defectos en el código que define los límites—las diferencias entre una partición equivalente y la que está justo por debajo o encima de ésta. En

otras palabras, ¿Muestra el programa un comportamiento Y cuando debería mostrar el comportamiento X, o piensa éste que está tratando con la partición de equivalencia B cuando éste debería reconocer esta situación como perteneciente a una partición de equivalencia A?

Ahora, note que sólo podemos utilizar el análisis de valores límite cuando los elementos de la partición de equivalencia son ordenados. En otras palabras, tenemos que ser capaces de hablar de un valor que es mayor que o menor que otro valor.

En la siguiente sección, examinaremos algunos ejemplos. Ilustraré el particionamiento de equivalencia y los valores límite para las pruebas funcionales. Sin embargo, tenga en cuenta que las particiones y los límites de equivalencia no funcionales, así como aquellas para las configuraciones, la capacidad del disco, el volumen de la red y así sucesivamente, pueden ser también utilizados para las pruebas no funcionales.

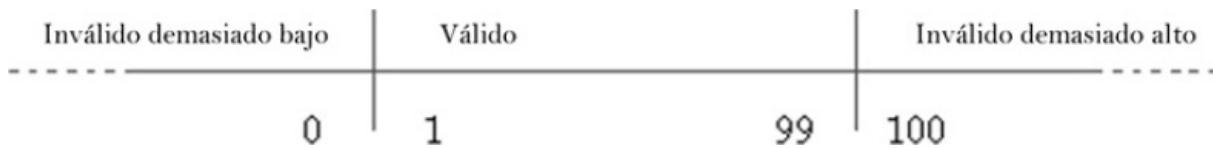


Figura 4.11: Entero

Examinemos las particiones de equivalencia y los valores límite para un campo entero. Suponga que usted está probando un sistema de comercio electrónico. Un campo que necesitaría probar es el que le permite a alguien especificar la cantidad de un ítem que quiere pedir. Supongamos que su sistema permitirá que alguien pida máximo hasta 99 de cualquier ítem dado, y que los ítems sean ítems indivisibles como libros, no ítems como la harina que se pueden vender en casi cualquier cantidad.

Esta regla significa que los valores del 1 hasta el 99 son la partición de equivalencia válida. Si es introducido un valor de 0 o menos, que es inválido, entonces aquellos valores forman la partición de equivalencia inválida demasiado baja. Si un valor de 100 o más es introducido, el cuál es inválido, entonces aquellos valores forman la partición de equivalencia inválida demasiado alta.

El valor 0 es el miembro más grande de la partición de equivalencia inválida demasiado baja, y por lo tanto es un valor límite. El valor 1 es el miembro más pequeño de la partición de equivalencia válida, y por lo tanto es un valor límite. El valor 99 es el miembro más grande de la partición de equivalencia válida, y por lo tanto es un valor límite.

El valor 100 es el miembro más pequeño de la partición de equivalencia inválida demasiado alta, y por lo tanto es también un valor límite.

Note que hay otros dos valores límite, que no se muestran aquí: el miembro más pequeño de la partición de equivalencia inválida demasiado baja y el miembro más grande de la partición de equivalencia inválida demasiado alta. Le podría ser difícil de identificar estos valores extremos, porque ellos dependen de la representación interna de los números enteros de la computadora, el número máximo de dígitos que puede ingresar en un campo, y otras restricciones parecidas. Sin embargo, debería hacer un esfuerzo para probarlos, porque los valores extremos son un lugar común donde se ocultan los defectos.

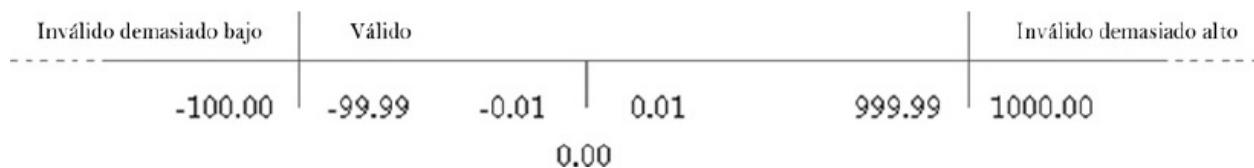


Figura 4.12: Número Real

Examinemos las particiones de equivalencia y los valores límite para un campo de un número decimal o real. Supongamos que se está probando una aplicación que calcula y muestra datos meteorológicos de varios tipos, incluyendo las temperaturas promedio para las ciudades seleccionadas. Como un probador, suponga que puede manipular la fuente de los datos en la base de datos de la aplicación para forzar ciertos cálculos y salidas interesantes.

Pruebe el cálculo y la presentación de las temperaturas promedio. Usted lee en la especificación que el campo debe ser exacto con la precisión de dos dígitos decimales, es decir, hasta las centésimas. La pantalla mostrará siempre dos dígitos decimales después del punto decimal, incluso si la parte decimal del valor es de 0,5 o 0,7 o incluso 0,0. En otras palabras, incluso si sólo hay uno o incluso ningún dígito decimal para mostrar. Además, la especificación dice que la pantalla acomodará seis caracteres en total, incluyendo el signo negativo para caracteres menores a cero.

Entonces, basados en esta especificación, el valor -100,00 es el miembro más grande de la partición de equivalencia inválida demasiada baja, y por lo tanto es un valor límite. El valor -99,99 es el miembro más pequeño de la partición de equivalencia válida, y por lo tanto es un valor límite.

El valor 999,99 es el miembro más grande de la partición de equivalencia válida, y por lo tanto es un valor límite. El valor 1.000,00 es el miembro más pequeño de la partición de equivalencia inválida demasiada alta, y por lo tanto es también un valor límite.

Para algunos esto podría ser suficientemente bueno, pero no para nosotros. Observe, la partición de equivalencia válida incluye tanto los números positivos como los negativos y el cero. Lo que sea que la especificación diga o no diga acerca del manejo diferente de estos valores, nosotros sabemos que los programadores cometan errores en tales situaciones—en parte porque nosotros solíamos trabajar como programadores y todavía programamos y sabemos que hemos cometido esos errores. Entonces, vamos a subparticionar la partición de equivalencia válida en tres subparticiones o subclases: válido negativo; válido cero, y válido positivo.

Entonces, el valor -99,99, ya identificado, es el miembro más pequeño de la equivalencia válida negativa, y por lo tanto es un valor límite. El valor -0,01 es el miembro más grande de la partición de equivalencia válida negativa, y por lo tanto es un valor límite. El valor 0,00 es a la vez el miembro más grande y el más pequeño de la partición de equivalencia válida cero—de hecho, es el único miembro—y por lo tanto es un valor límite. El valor 0,01 es el miembro más pequeño de la partición de equivalencia válida positiva, y por lo tanto es un valor límite. El valor 999,99, ya identificado, es el miembro más grande de la partición de equivalencia válida positiva, y por lo tanto es un valor límite.

¿Qué tan lejos podemos ir con el subparticionamiento? ¡Tanto así como lo necesitamos o queramos! Mientras sigan existiendo interesantes valores de pruebas, que se ocultan en una partición de equivalencia, podemos subparticionar esa partición para tratar de revelarlos.

Note, también, que en la figura 4.11 y la anterior, hemos dejado de lado toda una dimensión del particionamiento, la cual es para las entradas las cuales no son ni números enteros o ni decimales válidos en absoluto. Las entradas no numéricas para este campo decimal podría incluir letras, espacios en blanco, signos de puntuación, caracteres de control y nulos. Para un campo entero, podemos añadir números decimales en las entradas rechazadas, aunque ellos sean números.

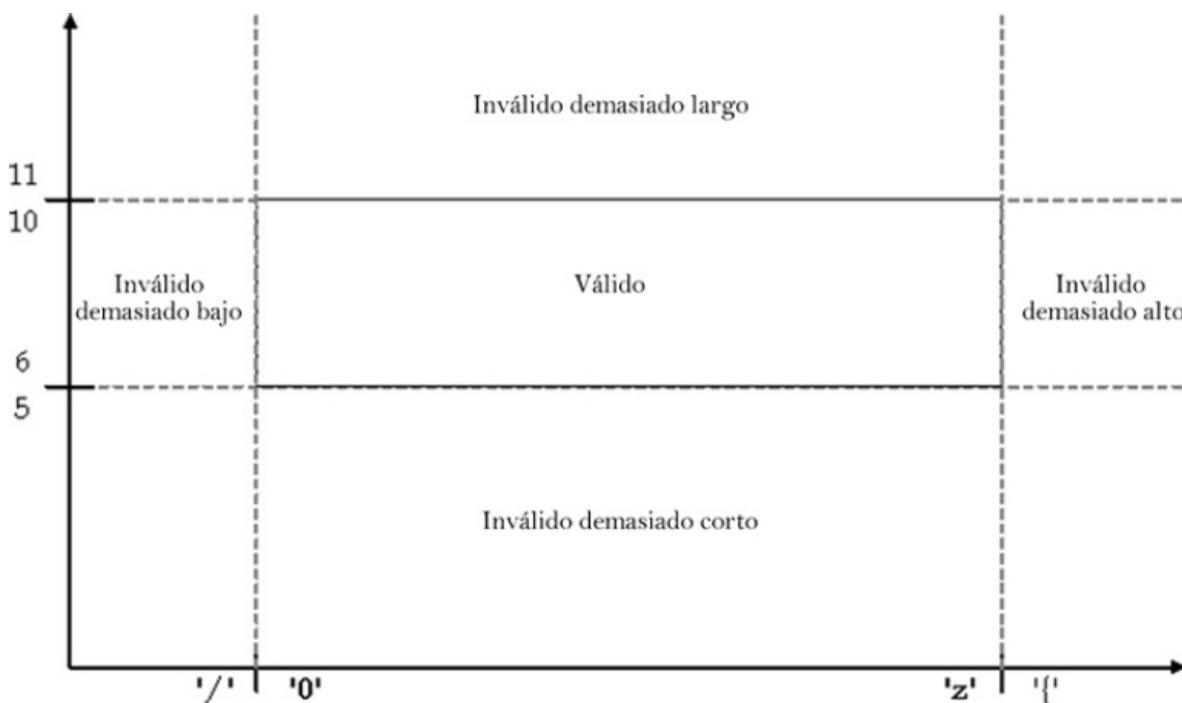


Figura 4.13: Carácter y Cadena de Texto

Esta argumentación acerca de las entradas más generales nos lleva al análisis de caracteres y cadenas de texto. Suponga que está probando una característica de seguridad que requiere contraseñas alfanuméricas de entre 6 y 10 caracteres de largo.

Podemos imaginar dos dimensiones del particionamiento, una basada en la longitud, una en los caracteres de la contraseña. Tenga en cuenta que esta figura mental crea una región válida en la gráfica, junto con cuatro regiones en las cuales la contraseña es inválida por exactamente una razón: longitud ilegal o caracteres ilegales.

Estas regiones inválidas “simples” están directamente a la izquierda y derecha y directamente arriba y abajo de la región válida.

También tenemos cuatro regiones en las cuales la contraseña es inválida por exactamente dos razones: longitud no permitida y caracteres no permitidos. ¿Necesitamos probar estas? Bueno, tal vez.

Como mencionamos antes, deberíamos probar solamente combinaciones de los inválidos cuando

vamos a ejecutar también pruebas que son inválidas por una razón **singular**. Y entonces, deberíamos hacerlo solo cuando pensamos que sirve para un propósito útil. ¿Cuál es la razón que hay para esperar problemas en la combinación de las inválidas, o hay alguna razón de negocios primordial para asegurar que éstas sean rechazadas correctamente?

Recuerde que las pruebas implican la selección de un subconjunto finito de pruebas que deben ser ejecutadas a partir de un conjunto infinito de pruebas que usted podría ejecutar. Debido a las restricciones de tiempo y dinero, la selección de una prueba significa por lo general la **deselección** de alguna otra prueba. Por lo tanto, antes de que salgamos corriendo hacia la gran nube infinita de pruebas que posiblemente podríamos ejecutar, haga memoria en su análisis de los riesgos y pregúntese usted mismo si el nivel de riesgo asociado con este caso de prueba justifica pruebas extensas.

DD/MM/YY

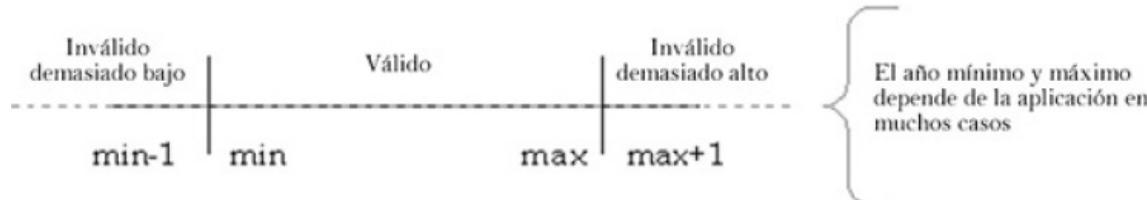
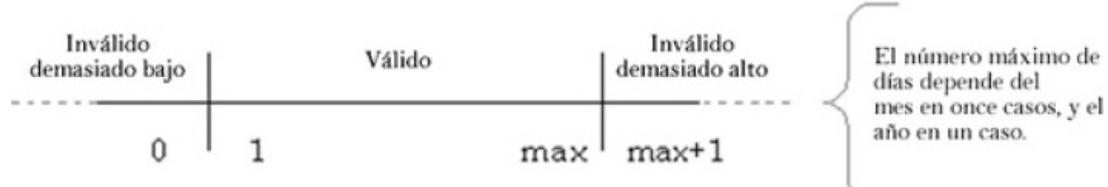


Figura 4.14: Fecha

Examinemos las particiones de equivalencia y los valores límite para un campo fecha. Suponga que usted está probando una aplicación web que le permite reservar pasajes de avión en línea.

Un campo que necesitaría probar es el que permite a alguien especificar la fecha de salida.

Las fechas son interesantes para el particionamiento de equivalencia y el análisis de valores límite porque hay dos dimensiones de validez que deben ser consideradas. Primero, ¿Es la fecha válida **como una fecha** en sí? Las fechas son campos que constan de tres subcampos. La validez de un subcampo—el día—depende de otro subcampo—el mes—durante once meses, y de dos subcampos—el mes y el año—para el duodécimo mes, Febrero.

Segundo, ¿Es la fecha válida en esta situación determinada? En otras palabras, ¿Es esta fecha válida tomando en cuenta lo que le estamos pidiendo al programa que haga con esta? Una fecha en el pasado, por ejemplo, no es una fecha válida, si estamos tratando de especificar una fecha de salida para un vuelo que queremos reservar. Por el contrario, una fecha en el futuro no es una fecha válida, si estamos tratando de especificar nuestra fecha de nacimiento.

HH:MM:SS

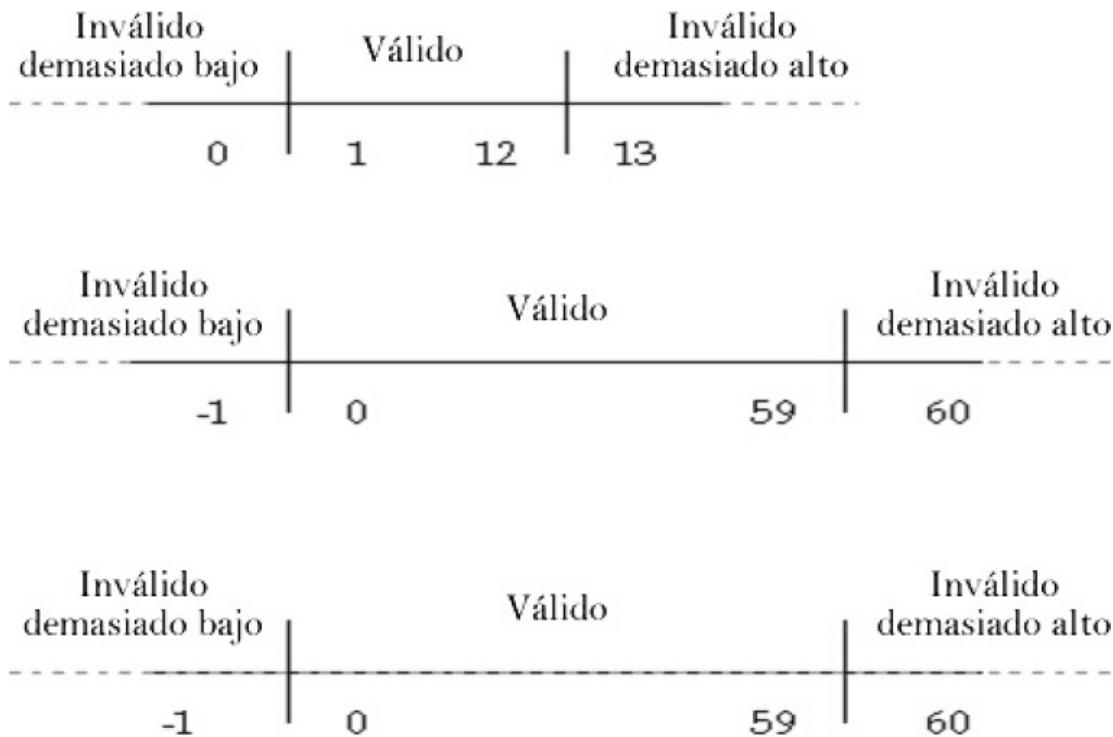


Figura 4.15: Hora

Continuando con nuestro sistema web de reserva de vuelos, suponga que examinamos un campo para el horario que nos pide la hora de salida preferida.

Una vez más, tenemos dos dimensiones interesantes de validez. La primera es el horario como un valor de tiempo. Los horarios, como las fechas, son campos que normalmente consisten en subcampos. El número de subcampos puede variar, por ejemplo, en este caso, usted probablemente no especificaría la hora de salida preferida incluyendo el subcampo de los segundos. De hecho, podría solamente especificar dos subcampos: la hora y AM o PM. El formato del campo para el horario—AM/PM o de 24 horas—infuye la validez del subcampo hora.

Al igual que con las fechas, un horario puede ser válido o inválido para una situación dada. Por ejemplo, no podemos salir en un horario que ya ha pasado. No se puede especificar un horario de salida deseado que está después de nuestro horario de llegada deseado, en la mayoría de los casos. Sin embargo, con los viajes en avión, si estamos volando a través de la línea internacional del cambio de fecha, podría haber ocasiones donde las fechas y los horarios de llegada preceden a las fechas y los horarios de salida.



Figura 4.16: Moneda

Finalmente, para las monedas, podemos identificar valores límite y particiones de equivalencia como si estuviéramos trabajando con números reales o decimales. Sin embargo, hay tres cosas importantes para recordar aquí.

Primero, mientras que el redondeo de los valores decimales puede ocurrir con números decimales, el redondeo de cantidades significativas de dinero es usualmente considerado un defecto. Es decir, si usted tiene cuenta de inversión con millones de dólares, generalmente esperaría que el balance de esa cuenta sea válido hasta la más pequeña unidad monetaria significativa. En los Estados Unidos y en una serie de otros países, eso sería centésimas de la moneda, p.ej., los centavos.

Para otros países, la unidad significativa más pequeña sería los veinteavos de la moneda; p.ej. en Nueva Zelanda e Israel, el dólar neozelandés y el shekel israelí son redondeados al lugar más cercano de un veinteavo en las transacciones en efectivo. Aún para otros países la unidad monetaria significativa más pequeña es la moneda misma, como el won coreano.

Segundo, cuando se trata de un programa que debe manejar monedas para muchos países, se presenta la pregunta del valor de la moneda. En el ejemplo que se muestra en esta figura, el precio máximo de oferta de una acción es 999.99. Eso es una cantidad significativa de dinero, si estuviéramos hablando de 999.99 libras, o de 999.99 euros, o incluso de 999.99 dólares. Sin embargo, si estuviéramos hablando 999.99 wons, eso es una cantidad muy pequeña de dinero. Los límites superiores e inferiores codificados en duro, no tendrían sentido en tales situaciones.

Tercero, si el sistema bajo prueba debe manejar cantidades de dinero históricamente, entonces tiene que tener cuidado con los supuestos acerca de la unidad significativa más pequeña de la moneda. Han pasado una serie de años desde que los ingleses adoptaron un sistema decimal para las libras, pero antes de eso, el sistema monetario inglés utilizaba libras, chelines y peniques. Los chelines eran una fracción no decimal de una libra y los peniques una fracción no decimal de un chelín. Es poco probable que encontraría esta situación con software moderno. Sin embargo, la transición a los precios de las acciones y bonos netamente decimales es un acontecimiento relativamente reciente en los Estados Unidos. Antes de esa transición, los precios de las acciones eran calculadas en octavos, y los precios de los bonos en 32avos.

Hasta ahora, hemos estado examinando las técnicas de diseño de pruebas que se aplican a entradas, salidas, valores de configuración individuales, y etc. Con las pruebas de caso de uso y pruebas de escenario empezamos a ver las pruebas de los flujos de trabajo completos, una pantalla completa o una serie de pantallas que logran una tarea determinada.

Los casos de uso son una técnica de diseño de sistemas utilizada a menudo con el diseño orientado a objetos. Para otros tipos de sistemas, la analogía es el escenario.

Cuando un caso de uso se está empleando—típicamente entregado a nosotros por nuestros colegas de desarrollo—o creando un escenario para las pruebas, por lo general diseñamos varios casos de prueba, que reflejan la utilización del mundo real típico o desafiante de la aplicación sometida a pruebas.

Por ejemplo, suponga que está probando una aplicación de préstamos hipotecarios. Un solicitante solicita tal préstamo. El solicitante es evaluado en función de su situación familiar, su valor de la garantía hipotecaria, sus activos, sus ingresos, su historial crediticio y el tamaño del préstamo el cual está solicitando. Entonces, para probar esto, necesitamos generar casos de prueba que llenen los campos de los datos pertinentes con estos valores de evaluación.

Podríamos crear un solicitante hipotético como sigue:

“John y Jenny Stevens, tienen tres hijos...”.

“... una casa con un valor de US\$ 400 mil de la cuál ellos deben US\$ 350 mil...”.

“... dos coches con un valor de US\$ 25 mil de lo cual ellos deben US\$ 17 mil...”.

“... ingresos de US\$ 45 mil y 75 mil, respectivamente...”.

“... una mora en el pago de su tarjeta Visa y una mora en el pago de los coches, diecisiete meses y treinta y cinco meses, respectivamente ..”.

“... y solicitan un préstamo hipotecario de US\$ 15 mil”.

Tendríamos también que determinar, basados en las reglas de negocio, dónde John y Jenny Stevens deberían ser aprobados para este préstamo.

Porque algunas metodologías de diseño orientado a objetos incluyen casos de uso, entonces esta puede ser una fuente fácil de pruebas. Todo lo que tenemos que hacer es generar los valores específicos y los resultados esperados.

Hablemos uno poco más acerca de los casos de uso, lo que son, cómo se ven y como utilizarlos.

Fundamentalmente, un caso de uso es una descripción de texto o gráfica de las interacciones que ocurren entre los que son denominados como “actores”. Los actores incluyen los usuarios, los clientes y el sistema. Un caso de uso debería producir algún resultado valioso, para uno o más actores, o para uno o más de los interesados del negocio en el sistema.

Los casos de uso pueden ser abstractos, en ciertos aspectos independientes del sistema mismo, así como un caso de uso de negocio o proceso de negocio, donde la tecnología es casi invisible. Sin embargo, los casos de uso también pueden ser muy concretos y específicos, describiendo un caso de uso del sistema en el nivel de la funcionalidad del sistema específico.

Cada caso de uso tiene precondiciones, las cuales deben ser verdaderas antes del primer paso del caso de uso, y cuál debe ser verdadero para que el caso de uso continúe satisfactoriamente. Cada caso de uso tiene poscondiciones también, las cuales son los resultados observables y el estado final del sistema después de que los pasos del caso de uso hayan terminado.

Un caso de uso tiene usualmente un camino del flujo principal, el flujo de trabajo o el escenario. Estos son el conjunto de pasos más probables que ocurren, algunas veces denominados el camino feliz. Adicionalmente, el caso de uso debería tener caminos alternativos definidos para los problemas o las variaciones típicas previsibles.

En esta página, vemos un ejemplo de un caso de uso informal, que describe las compras de un sitio de comercio electrónico, de la tienda del sitio web de RBCS. En la parte superior, tenemos el flujo de trabajo normal de la compra del sitio web. Éste es un camino feliz. Note que la precondición es de que el cliente—quién es uno de los actores en este caso de uso—está en la tienda en el sitio web de RBCS—el cuál es el otro actor en este caso de uso—navegando a través de los cursos, libros y otros ítems en venta allí. Primero, el Cliente coloca uno o más ítems en su carro de compras. Luego el Cliente selecciona “pagar”. Ahora, el sistema recopila la dirección, el pago y la información del envío del Cliente. Con la información recopilada, el Sistema muestra toda la información para la

confirmación por parte del Cliente. Finalmente, el Cliente confirma el pedido al sistema para que sea entregado.

Note que el resultado final, la poscondición, es que el pedido está en el sistema para ser entregado. Presumiblemente otro caso de uso para completar el pedido describirá cómo este pedido llega en última instancia al hogar o lugar de trabajo del cliente.

También vemos algunos flujos de trabajos definidos con excepciones.

Por un lado, el Cliente podría tratar de pasar a pagar con un carro de compras vacío. En ese caso, el Sistema da un mensaje de error.

Por otro lado, el Cliente podría proporcionar la información inválida de la dirección, el pago, o el envío. En cada pantalla—si estuviéramos siguiendo el flujo típico del comercio electrónico—el Sistema da un mensaje de error como sea apropiado y bloquea cualquier proceso adicional hasta que los errores sean resueltos.

Finalmente, el Cliente podría abandonar la transacción antes o durante el paso del pago. Para controlar esto, el Sistema retira el Cliente del sistema después de 10 minutos de inactividad.

Note que para derivar las pruebas para este caso de uso, usted selecciona simplemente los datos específicos y crea los casos de prueba que corresponden al camino feliz y a los flujos de trabajo excepcionales. Típicamente, usted ejecutaría una prueba por lo menos por cada flujo de trabajo. Los flujos de trabajo riesgosos merecerían, por supuesto, más de una prueba. Un número importante de particiones de equivalencia—p.ej. diferentes tarjetas de crédito permitidas en este ejemplo—conducirían a múltiples pruebas.

Debería mencionar también que la precondición, la poscondición y los actores están implícitos en este caso de uso en vez de que sean especificados explícitamente. Es típico para los casos de uso informales de dejar algunos de los elementos de un caso de uso formal. Otros elementos de un caso de uso formal no vistos aquí son: un número de Identificación (p.ej. para la trazabilidad), un nombre, una descripción, una prioridad y una frecuencia estimada o medida de uso. En un caso de uso formal, todos estos elementos serían especificados explícitamente en una cabecera de un caso de uso literal o en un bloque de texto en un caso de uso gráfico.

Está bien combinar técnicas de diseño de pruebas—de hecho, usted debería realmente—pero en algunos casos, cosas extrañas pueden ocurrir. Por ejemplo, necesita diseñar pruebas basadas en los casos de uso o escenarios con las particiones de equivalencia. Sin embargo, ¿Debería utilizar los valores límite?

El utilizar los valores límite podría llevarnos a crear pruebas que no son necesariamente el uso razonable del sistema. Por ejemplo, si la aplicación utiliza un elemento sin signo de cuatro bytes como una variable contador, ¿debería probar ordenando millones de ítems sólo porque el software lo soporta? O ¿debería el software imponer un límite superior razonable en la cantidad del pedido? Si es así, ¿cuál es ese límite? ¿Está especificado?

Como mencionamos antes, las suposiciones aparentemente razonables acerca de los límites lógicos no podrían aplicarse en todas las situaciones. En ciertos casos, basados en el horario local, un sistema de reserva de vuelos podría permitir para los viajes, fechas de salida posteriores a las fechas de llegada.

Los casos de uso y escenarios pueden a menudo revelar el hecho de que, con frecuencia, las especificaciones de los límites podrían no estar completas. En algunos casos, aunque el sistema pueda manejarlo, aceptar una entrada “ridícula” constituye un defecto.

Con las tablas de decisión, continuamos con las pruebas de completos flujos de trabajo, una pantalla entera o una serie de pantallas que logran una tarea determinada. Las tablas de decisión son una forma agradable, compacta y fácil de leer para expresar las reglas de negocio que determinan como una aplicación debería manejar estos flujos de trabajo.

Por ejemplo, una tabla de decisión nos puede decir cómo el sistema debería procesar un pedido basado en el tamaño, el stock disponible, el estado en el cuál se puede enviar, y así sucesivamente. Una tabla de decisión es una representación tabulada de las reglas, la lógica de negocios. Estas mismas reglas pueden ser mostradas como diagramas de flujo, pero eso no es algo que vamos a cubrir en este libro.

Si usted es un probador, la obtención de las tablas de decisión es bonito, porque las puede utilizar para crear casos de prueba muy fácilmente. Si recibe un diagrama de flujo o una descripción de texto de las reglas de negocio, la creación de una tabla de decisión de esas reglas puede también hacer fácil las pruebas.

Veamos un ejemplo.

<i>Condición</i>	1	2	3	4	5
Tarjeta Válida	No	Si	Si	Si	Si
PIN Válido	-	No	No	Si	Si
PIN Inválido=3	-	No	Si	No	No
Balance OK	-	-	-	No	Si
<i>Acción</i>					
Rechazar Tarjeta	Si	No	No	No	No
Reingresar PIN	No	Si	No	No	No
Guardar Tarjeta	No	No	Si	No	No
Reingresar Pedido	No	No	No	Si	No
Dispensar efectivo	No	No	No	No	Si

Tabla 4.9: Tabla de Decisiones de un Cajero Automático

Esta tabla muestra una tabla de decisión de un cajero automático (ATM), que sólo se ocupa de solicitudes de retiro. Muestra el aspecto típico de una tabla de decisión. En el lado izquierdo hay una columna con las condiciones enumeradas en la parte superior y las acciones en la parte inferior. En cada columna de la derecha, hay una regla la cual establece cuáles acciones son y no son tomadas basadas en cada una de las condiciones que se han cumplido o no.

La primera regla dice que, si alguien inserta una tarjeta inválida en el cajero automático, el cajero automático debería rechazar esa tarjeta. No debería solicitar al usuario que introduzca o vuelva a introducir el PIN, porque ningún PIN aplica para tal tarjeta. No debería retener la tarjeta. No debería solicitar al usuario que ingrese o vuelva a ingresar el monto del retiro solicitado porque ninguna cuenta está asociada a la tarjeta. ¡Ciertamente, no debería dar efectivo!

La segunda regla dice que, si alguien inserta una tarjeta válida en el cajero automático, pero introduce un PIN inválido, el cajero automático debería solicitar al usuario que vuelva a introducir el PIN, con la condición de que no hayan ingresado ya un PIN inválido tres veces. La tercera regla aclara esta situación diciendo que, si el usuario introduce un PIN inválido tres veces seguidas, debería retener la tarjeta.

La cuarta regla dice que, si alguien inserta una tarjeta válida en el cajero automático e ingresa un PIN válido, pero una solicitud inválida de retiro, el cajero automático debería solicitar al usuario que vuelva a introducir la solicitud de retiro.

La quinta regla—el llamado “camino feliz”—dice que, dada una tarjeta, el PIN, y la solicitud válida de retiro, el cajero automático debería dar efectivo.

Esta tabla de decisión muestra la lógica de negocios para un cajero automático. Observe que los guiones “-” indican condiciones que no son alcanzadas o no son aplicables para esta regla. Las reglas son mutuamente exclusivas, porque sólo una regla puede aplicarse en un momento determinado en el instante.

Observe que la capa de la lógica de negocios está generalmente bajo la capa de la interfaz de usuario, de modo que en este punto el control de la sanidad básica de las entradas debería haber sido realizado. En otras palabras, los valores límite y las particiones de equivalencia inválidas de los campos de entrada deberían ser utilizados principalmente para probar la validación de las entradas. Una vez que consigamos probar la lógica de negocios, no deberíamos tener necesidad de repetir aquella validación de las entradas.

Cuando realiza pruebas con una tabla de decisión, la regla usual de la cobertura es tener por lo menos una prueba por cada regla de negocios o columna en la tabla de decisión. Decimos “por lo menos una prueba” porque, en algunas situaciones, más que una manera interesante puede existir para satisfacer o no una condición y que sea valiosa de probar. Por ejemplo, con nuestras pruebas del Cajero Automático, podríamos querer probar las tarjetas del cajero automático vencidas, las tarjetas del Cajero Automático que utilizan una red de efectivo no compatible, las tarjetas del Cajero Automático robadas, las licencias de conducir u otras tarjetas con una banda magnética y las tarjetas sin banda magnética. Tenga en cuenta que ésta es una aplicación de la técnica de particionamiento de equivalencia para una sola columna de la tabla de decisión.

En la próxima tabla, examinaremos una tabla de decisión más compleja.

Hace años, cuando Rex Black solía conducir un poco más rápido de lo que debía, se encontró con un policía que utilizó una computadora de mano para gestionar su notificación por el exceso de velocidad. Esta computadora también le permitía al policía aceptar su tarjeta de crédito para el pago de la multa—una característica práctica, porque le impidió perder el vuelo de la aerolínea que estaba tratando de tomar con prontitud.

Como esto fue hace unos cuantos años atrás, ahora incluso más departamentos de policía cuentan con computadoras de mano que emiten notificaciones e incluso aceptan tarjetas de crédito para pagar las multas por las violaciones de conducción y otras infracciones.

Una tabla de decisión podría ser utilizada para diseñar y probar tal sistema. Las condiciones que determinan las acciones que deben ser tomadas son mostradas en la tabla de decisión en la siguiente tabla. En esta situación, puede que quiera utilizar el análisis de valores límite para cubrir adecuadamente las reglas. Además, tendremos que considerar cómo administrar la interacción de las reglas.

<i>Condición</i>	1	2	3	4	5	6	7	8
Licencia OK	No	-	-	-	-	-	-	-
Autorización	-	Si	-	-	-	-	-	-
Registro OK	-	-	No	-	-	-	-	-
Vehículo OK	-	-	-	No	-	-	-	-
Exceso de velocidad	-	-	-	-	1-10	11-20	21-25	>25
<hr/>								
<i>Acción</i>								
Arresto	Si	Si	-	-	-	-	-	Si
Ticket de infracción	-	-	Si	Si	-	-	-	-
Advertencia	-	-	-	-	Si	-	-	-
Multa	+250	+250	+25	+25	+0	+75	+150	+250

Tabla 4.10: La Tabla de Decisiones de un Sistema de la Policía

Aquí tenemos una tabla de decisión—o más probablemente una parte de una tabla de decisión—que podría describir algunas de las acciones de tal computadora de la policía.

La primera regla dice que si el conductor tiene una licencia de conducir inválida, el conductor será arrestado. El conductor incurrió también en una multa de US\$ 250. Las líneas punteadas aquí nos dicen que otras condiciones y acciones podrían aplicarse también.

La segunda regla dice que si el conductor tiene una orden judicial existente para su arresto, el conductor será arrestado. El conductor incurrió también en una multa de US\$ 250. Una vez más, otras condiciones y acciones se podrían aplicar.

La tercera regla dice que si el conductor no ha pagado sus cuotas del registro, el conductor debe recibir una boleta de infracción corregible que obliga al conductor a pagar la cuota en algún plazo determinado. El conductor incurrió también en una multa de US\$ 25. Si el conductor no paga sus cuotas, típicamente, esa notificación expirará y se convertirá en una orden judicial de arresto.

La cuarta regla dice similarmente que si el conductor tiene un vehículo con desperfectos, el conductor debería recibir una boleta corregible que requiere que el conductor resuelva el problema dentro de algún plazo determinado. El conductor incurrió también en una multa de US\$ 25. Si el conductor no resuelve el problema, típicamente esa notificación expirará y se convertirá en una orden judicial de arresto.

Cada una del primer conjunto de las cuatro reglas se puede aplicar simultáneamente. Es decir, estas reglas no son mutuamente exclusivas. Los totales de multa serían adicionados en estas situaciones.

La quinta regla dice que si el conductor estaba viajando entre 1 y 10 millas por hora (o km/hora si lo prefiere) sobre el límite de velocidad, el conductor recibirá solo una advertencia.

La sexta regla dice que si el conductor estaba viajando entre 11 y 20 millas por hora (o km/hora si lo prefiere) sobre el límite de velocidad, el conductor incurrió en una multa de US\$ 75.

La séptima regla dice que si el conductor estaba viajando entre 21 y 25 millas por hora (o km/hora si lo prefiere) sobre el límite de velocidad, el conductor incurrió en una multa de US\$ 150.

La octava regla dice que si el conductor estaba viajando a más de 25 millas por hora (o km/hora si lo prefiere) sobre el límite de velocidad, el conductor incurrirá en una multa de US\$ 250. El conductor será arrestado entonces.

Todas estas reglas del segundo conjunto, que consisten en 4 reglas, son mutuamente exclusivas.

Tome en cuenta cuánto más tiempo es necesario para en la descripción de estas reglas verbalmente o textualmente. Usted puede observar cuán compacta y aún sin ambigüedades puede ser la representación de la tabla de decisión de las reglas de negocio.

Tenga en cuenta que la multa máxima puede ser US\$ 800, basada en la interacción de las cuatro primeras reglas y la octava regla. ¡Por supuesto, el conductor será arrestado solo una vez!

Cuando se prueba una tabla de decisión como esta, primero tenemos que probar cada regla por sí misma. Para las reglas definidas en intervalos, como las reglas de la cinco a la ocho, probablemente necesitaremos dos pruebas para asegurar que los rangos son definidos correctamente en los límites.

Una vez que hayamos probado las reglas por sí mismas, deberíamos entonces probar las combinaciones de las reglas. Hay técnicas para decidir cómo realizarlo así, tales como los árboles de clasificación, las tablas de todos los pares y el análisis de dominio. Estas técnicas son descritas en el libro *Advanced Software Testing for the Test Analyst*.

Las tablas de decisión son buenas cuando usted está probando un sistema que controla transacciones individuales o lotes de entradas, y cada transacción o lote es manejado de forma independiente. Sin embargo, para algunos sistemas las series de eventos y condiciones que han ocurrido parcialmente en el pasado, determinan cómo el sistema responderá al evento actual y a las condiciones que predominan. Para las pruebas de estos sistemas deberíamos utilizar los diagramas de transición de estados. Para crear un diagrama de transición de estados, usted primero identifica los varios estados en los que el sistema puede estar. Querrá asegurarse de que entiende los estados inicial y final. Como regla general, los casos de prueba deberían comenzar con el sistema en el estado inicial y deben terminar con el sistema en el estado final.

En cada estado, algún evento puede ocurrir que forzará una transición de estado. (Esto es cierto para todos los estados excepto el estado final.). Mientras la transición de estado ocurre, el sistema podría tomar alguna acción. En algunos casos, dos o más condiciones son aplicadas al evento, el cual influye cuál transición ocurre—y por lo tanto cuál acción es tomada. Después que la transición de estado ha ocurrido, el sistema está ya sea en un nuevo estado o se ha mantenido en su estado actual.

Podemos representar el comportamiento del sistema ya sea en un gráfico o en una tabla. El gráfico o tabla sirve como un modelo del sistema. Como tal, también puede servir como un oráculo para nuestras pruebas, ayudándonos a determinar la acción correcta y el nuevo estado asociado con un estado actual particular y un par evento/condición.

Veamos un ejemplo.



Un diagrama transiciones de estados para un sistema de punto de venta, desde el punto de vista de un cajero. ¿Cómo se vería desde el punto de vista del cliente? ¿Del sistema? ¿Ve el defecto?

Figura 4.17: Diagrama de Transiciones de Estados

Esta figura muestra un diagrama de transición de estados para un sistema de punto de venta, igual al que encontraría en un supermercado o pulperia.

Hay por lo menos dos defectos en este diagrama. Primero, el cajero sólo puede terminar su turno

mientras acepta el pago del cliente. Esto es doblemente absurdo. Un cajero no debería dejar el trabajo en medio de la atención a un cliente. El cajero no debería tener que estar en medio de la atención a un cliente para dejar el trabajo.

Segundo, no hay ningún mecanismo para el abandono de una transacción si el cliente no puede hacer un pago válido.

Una de las cosas buenas acerca del diagrama de transición de estados es que hace que sea fácil de detectar problemas como éste. Esos problemas pueden esconderse en una descripción larga de texto de los requisitos.

Como con las tablas de decisión, la representación compacta y precisa ayuda a revelar los defectos y así acentúa el potencial de la prevención de los defectos por medio de las pruebas—si el diseño de pruebas es realizado antes que el sistema sea construido, ¡eso es! Y si usted persigue una estrategia de pruebas netamente reactiva, entonces perderá este beneficio.

Note que se muestra el sistema desde el punto de vista del cajero. Le recomendamos que vuelva a dibujar el diagrama desde el punto de vista del cliente. Luego, vuelva a dibujarlo de nuevo desde el punto de vista del propio sistema de punto de venta.

Cuando realizamos ejercicios en nuestras clases presenciales, algunas veces la gente se confunde acerca de la diferencia entre un estado y evento, o entre un estado y una acción. Un estado es una situación que persiste, y persistirá hasta que algún evento ocurra para causar una transición de estado. Un evento es un acontecimiento, algo que concluye casi al instante o en algún período de tiempo fijo, bastante constante. Los eventos son entradas frecuentes de algún tipo de sistema o aplicación.

Las acciones son también acontecimientos, pero ellas son usualmente salidas de algún tipo de sistema o aplicación. Típicamente, una acción se inicia y termina rápidamente o en algún período de tiempo fijo, relativamente constante. En algunos casos, encontrará un tanto arbitrario en cuanto a si elegir el modelado de un determinado comportamiento como una acción asociada a una transición de estado o como un estado intermedio con una transición de estado y acción asociados.

Como con las otras técnicas formales del diseño de pruebas, hay criterios de cobertura para los diagramas de transición de estados. Los criterios mínimos son: visitar cada estado y atravesar cada transición. Note que si atraviesa cada transición, habrá ejercido cada par de evento/condición en el diagrama así como también habrá observado cada acción en el diagrama, porque las transiciones de estado son activadas por los pares evento/condición y son tomadas acciones por el sistema durante las transiciones de estado.

Criterios más fuertes existen para la cobertura de los diagramas de transición de estados. Por ejemplo, el criterio de la cobertura de transiciones (“switch-coverage”) de Chow requiere la cobertura de las secuencias de las transiciones de longitud creciente. Estos criterios son tratados en el libro *Advanced Software Testing for the Test Analyst*.

Estado Actual	Evento[Condición]	Acción	Nuevo Estado
Ingresando al sistema	Clave[inválida]	Error	Ingresando al Sistema
Ingresando al sistema	Clave[válida]	Abrir registro	Esperando
Ingresando al sistema	Cliente	[Indefinido]	[Indefinido]
Ingresando al sistema	Escanear[cualquiera]	[Indefinido]	[Indefinido]
Ingresando al sistema	Pagar[cualquiera]	[Indefinido]	[Indefinido]
Ingresando al sistema	Shift+End	[Indefinido]	[Indefinido]

Tabla 4.11: Tabla de Transiciones de Estado

Los diagramas de transición de estados son bonitos, pero sólo muestran cómo el sistema maneja los pares evento/condición para ciertos estados. ¿Qué pasa si un evento ocurre cuando el sistema no lo está esperando? ¿Qué debería ocurrir?

Una tabla de estado-transición puede revelar situaciones indefinidas, como lo hace esta parte de la tabla para el gráfico anterior. Puede construir estas tablas del diagrama de transición de estados utilizando el siguiente proceso:

1. Haga una lista de todos los estados en el diagrama.
2. Haga una lista de todos los pares evento/condición en el diagrama.
3. Crear una tabla con cada combinación posible del estado con el par evento/condición.
4. Para cada combinación de estado-evento/condición, si el diagrama especifica la acción que debe

ser tomada y el nuevo estado al que se debe ingresar, ponga aquella información en la fila correspondiente en la tabla. Si no, escriba “indefinido” en aquellos dos campos en esa fila.

En este punto, puede haber encontrado algunos problemas de diseño del sistema. Para las filas con las acciones y los nuevos estados indefinidos, quizás desee comprobar con la gente para ver si aquellas son realmente situaciones las cuales no pueden suceder. Si no, entonces ya sea el programador se acordará de codificarlos durante la implementación—la cual es una suposición riesgosa— o el sistema tendrá un defecto de omisión que puede resultar en un comportamiento impredecible durante su utilización.

Una vez más, como todos los demás modelos formales para el diseño de pruebas, tenemos un criterio de cobertura para las tablas de transición de estados. Cada fila debería tener una prueba asociada—si por ninguna otra razón se confirma que las cosas que “no puede ocurrir” realmente no suceden.

Veamos otro ejemplo acerca de un diagrama de transición de estados.

En la siguiente página, verá un diagrama de transición de estados para un servidor de impresión. Suponga que este servidor de impresión es una aplicación que corre en una tarjeta independiente instalada en la misma impresora. El servidor de impresión puede estar en uno de los siguientes cinco estados:

1. Esperando por la tarea de impresión—el estado inicial e inactivo.
2. Poniendo la tarea en la cola de impresión.
3. Imprimiendo el trabajo de impresión.
4. Esperando la intervención del usuario para terminar el trabajo de impresión.
5. Esperando la intervención del operador para restaurar la impresora en un estado de funcionamiento.

El servidor de impresión responde a los eventos, los cuales son entradas, ya sea de los usuarios o la impresora, basado en el estado en el que éste esté.

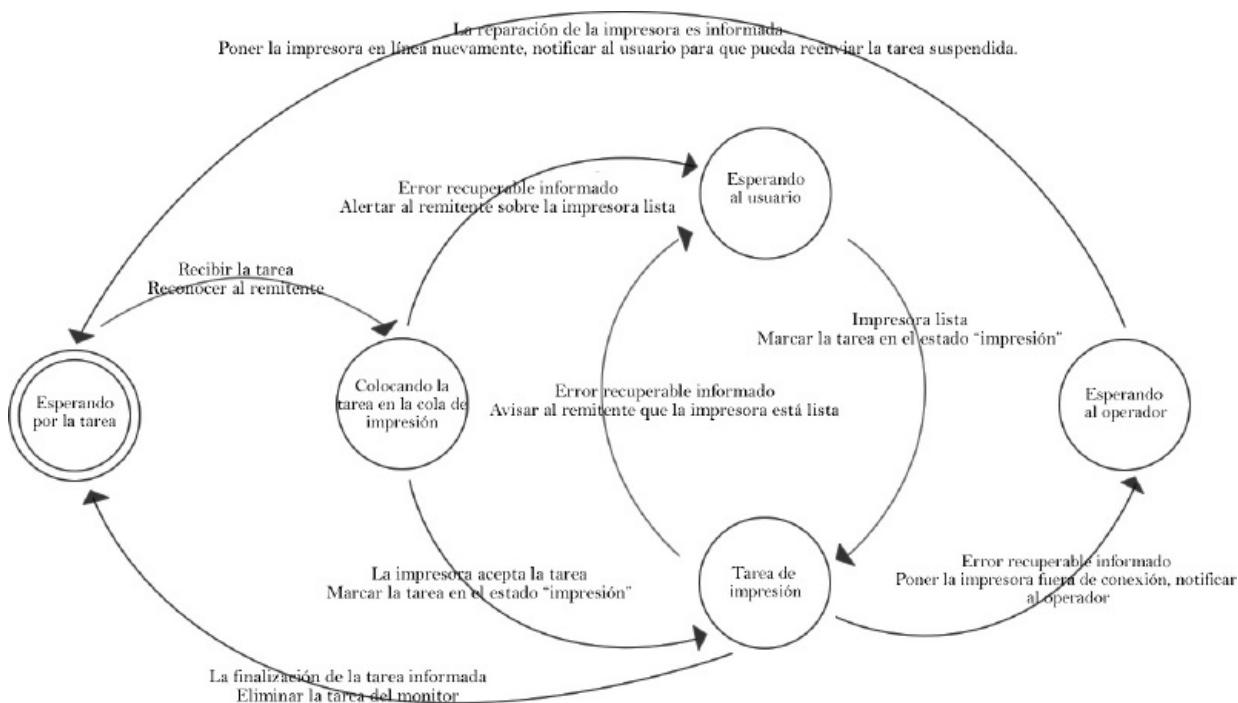


Figura 4.18: El Gráfico de la Máquina de Estados del Servidor de Impresión

Aquí está el diagrama de transición de estados para el servidor de impresión. Tómese un momento para estudiarlo.

Cree un conjunto de casos de prueba para este diagrama. Suponga que todas las pruebas deben comenzar en el estado inicial—mostrado con un círculo con borde doble—and deben terminar cuando las transiciones del sistema vuelvan al estado inicial. Trate de crear el número mínimo de casos de prueba y aún así lograr la cobertura completa de los estados y las transiciones de estados. ¿Terminó? Bien. Debería haber creado dos casos de prueba.

Teniendo en cuenta la regla acerca de los casos de prueba que tienen que empezar y terminar en el estado inicial, ¿Cuál es el número máximo de casos de prueba?

Es infinito. Si usted nos muestra cualquier conjunto de pruebas que piensa que es exhaustivo, podemos añadir otro ciclo del trabajo desde el trabajo de impresión a la espera de los usuarios y de vuelta al trabajo de impresión que es una nueva prueba. Esta propiedad, de siempre ser capaz de añadir una única variante más a una lista, es la definición misma del infinito contable.

Para cualquier sistema que puede describirse mediante un diagrama de transición de estados con esos bucles en el diagrama, puede crear un número infinito de pruebas.

Ahora cree una tabla de transición de estados.

¿Terminó? Bien. Su tabla de transición de estados debería tener 35 filas en la tabla. ¿Tiene 40? Si es así, probablemente contó dos veces el evento “error recuperable reportado”.

4.4 Técnicas Basadas en la Estructura

Objetivos del Aprendizaje

LO-4.4.1 Describir el concepto y el valor de cobertura de código. (K2)

LO-4.4.2 Explicar los conceptos de cobertura de sentencia y decisión, y dar las razones por qué estos conceptos pueden también ser utilizados en otros niveles de prueba distintos a las pruebas de componente (p.ej. en procedimientos comerciales en el nivel de sistema). (K2)

LO-4.4.3 Escribir casos de prueba de los flujos de control dados utilizando las técnicas de diseño de pruebas de sentencia y decisión. (K3)

LO-4.4.4 Evaluar la completitud de la cobertura de sentencia y decisión con respecto a los criterios de salida definidos. (K4)

En esta sección, Técnicas Basadas en la Estructura, cubrirá los siguientes conceptos clave:

- Niveles de cobertura de código.
- Obtención de la cobertura de sentencia y decisión con las pruebas.
- Utilización del flujo de control del programa para diseñar las pruebas.

Emparemos con los conceptos básicos de las pruebas basadas en la estructura o de caja blanca.

Recuerde que las pruebas basadas en la especificación, también conocidas como las pruebas de caja negra o como las pruebas de comportamiento, son las que son diseñadas desde la especificación del sistema. En las pruebas de comportamiento, nosotros ignoramos el funcionamiento interno del sistema y nos enfocamos acerca de cómo este se supone que se debe comportar.

Ahora, las pruebas basadas en la estructura, las cuales son llamadas también pruebas estructurales o de caja blanca, se basan en la estructura interna del sistema o en un componente del sistema; p.ej., nosotros examinamos cómo el sistema funciona y qué hace. Específicamente, seleccionamos las entradas, las precondiciones, las condiciones, los eventos u otros estímulos basados en una parte de la estructura del sistema que estos ejercerán.

Por supuesto, cuando realicemos las pruebas estructurales—porque éstas son pruebas dinámicas—el sistema o componente presentará algún comportamiento. Tenemos que comparar ese comportamiento con algún resultado esperado, de lo contrario no estamos probando. No podemos derivar los resultados esperados para las pruebas estructurales por completo de la estructura, porque de lo contrario terminamos probando el compilador, el sistema operativo y otros elementos del entorno, más que el sistema propio. De este modo nosotros derivamos típicamente los resultados esperados para las pruebas estructurales en parte de la estructura, pero también de las especificaciones, las expectativas razonables, y así sucesivamente.

Como con las pruebas de comportamiento, cuando nosotros estamos observando los comportamientos mostrados por una prueba de estructura, nosotros podemos estar observando comportamientos funcionales—¿Qué hace el sistema o componente? —O comportamientos no funcionales—¿Cómo lo hace?—basado en la clasificación del ISO 9126.

Hay tres maneras para diseñar las pruebas estructurales.

La más simple es analizar los flujos de control en el código y utilizar ese análisis para medir la cobertura de las pruebas existentes y para adicionar las pruebas adicionales para alcanzar un nivel deseado de cobertura. (Hablaremos más acerca de la cobertura de código más adelante). Estos tipos de pruebas estructurales son cubiertos en el programa de estudios básico.

La siguiente, de alguna manera más complicada de diseñar las pruebas estructurales es la de analizar los flujos de datos, utilizando el código y las estructuras de datos. Usted luego utiliza ese análisis para medir la cobertura de las pruebas existentes y para adicionar pruebas adicionales para alcanzar un nivel deseado de cobertura del flujo de datos. Estos tipos de pruebas estructurales **no** son cubiertos en el programa de estudios nivel básico.

La final, la manera más complicada de diseñar las pruebas estructurales es el de analizar las interfaces, las clases, los flujos de llamada y otros por el estilo observando las interfaces de programación de la aplicaciones (APIs); el hardware, el software y los documentos de diseño y los diagramas de un sistema de redes; las tablas de base de datos, las restricciones de integridad y los procedimientos memorizados (“stored procedures”). Como antes, usted puede analizar ese análisis para medir la cobertura de las pruebas existentes y para adicionar pruebas adicionales para alcanzar un nivel deseado de cobertura del diseño. Este tipo de diseño de pruebas estructurales es comúnmente utilizado durante las pruebas de integración y de sistema. Sin embargo, estos tipos de pruebas estructurales **no** son cubiertos en el programa de estudios básico.

Permítanos resaltar de nuevo un uso importante del diseño de pruebas estructural: Es una mejor práctica para medir el grado de cobertura estructural de las pruebas basadas en la especificación y la experiencia, de tal manera que usted pueda buscar partes importantes de la estructura del sistema, las cuales no están siendo probadas.

Si no tiene una base en programación, podría considerar esta sección un poco difícil. Porque comprendiendo cómo los sistemas tienden a fallar y por qué, es una habilidad clave cuando se está haciendo el análisis de los riesgos de calidad, nosotros lo animaríamos a que se auto enseñe un lenguaje de programación si intenta que su profesión sea la de pruebas de software.

Entonces, ¿Qué significa la cobertura de flujos de control en el código-más frecuentemente llamada cobertura de código-? ¿Qué niveles de cobertura de código existen?

Hay un número de variantes aquí, pero las más comunes son las siguientes:

La cobertura de sentencia, la cuál es el porcentaje de las sentencias ejecutadas mínimamente una vez por las pruebas.

La cobertura de rama o de decisión, la cuál es el porcentaje de ramas o decisiones ejecutadas mínimamente una vez por las pruebas. De nuevo, si usted dice que ha alcanzado "cobertura de rama" eso significa usualmente de que usted ha probado el 100% de las ramas. La cobertura de rama y decisión significan la misma cosa. La cobertura de rama se refiere a una manera gráfica de visualizar el flujo de control, mientras que la cobertura de decisión se enfoca en la condición, si simple o compuesta, que determina cuál rama será tomada.

Glosario del ISTQB

Cobertura de decisión: El porcentaje de resultados de decisión que hayan sido ejercidos por un juego de pruebas. 100% de cobertura de decisión implica ambos 100% de cobertura de rama y 100% cobertura de sentencia.

La ramificación ocurre cuando el programa hace una decisión acerca de que si una situación particular de dos o más situaciones posibles existen, y luego procede a fluir el control del programa en una manera apropiada para manejar la situación. Ya que una manera de manejar la situación es a través de la secuencia de sentencias, el 100% de cobertura de rama significa que alcanzaremos el 100% de cobertura de sentencia.

La cobertura de condición, la cuál es el porcentaje de condiciones simples, que han sido evaluadas por una de las pruebas mínimamente. Una condición simple es algo como "edad >= 18" o "nombre== "Robert"". El 100% de la cobertura de condición requiere que cada condición única en cada sentencia de decisión sea probada como verdadera y como falsa.

Ahora, cuando no sólo contamos con condiciones simples en un programa, pero con condiciones compuestas como "(edad >= 0) && (edad < 18)", entonces podemos hablar acerca de la cobertura de multicondición. La cobertura de multicondición (más conocida como la cobertura completa de condición múltiple) es el porcentaje de las condiciones de todos los resultados de cada condición única dentro de una sentencia que ha sido evaluada por mínimamente una de las pruebas.

Porque probando todas las combinaciones de las condiciones, también significa la prueba de todas las condiciones posibles, el 100% de la cobertura de condición múltiple implica el 100% de cobertura de condición.

Ahora, la cobertura de condición y decisión modificada (MC/DC) es una variante leve de la cobertura de multicondición. En la cobertura de condición y decisión modificada nos fijamos en el porcentaje de las combinaciones de las condiciones que pueden influenciar la decisión que ha sido probada. Por ejemplo, considere una condición compuesta como "(edad >= 0) && (edad < 18)". Si la primera condición es verdadera y el valor de la edad es menor que cero, entonces la condición compuesta completa será evaluada como falsa. No hay ninguna necesidad de tratar de evaluar la segunda condición, porque esta no influenciará la decisión. Muchos compiladores, incluyendo todos los compiladores de C++, cesarán la evaluación de una condición compuesta tan pronto como la decisión total sea conocida. Porque la prueba de todas las combinaciones de las condiciones es una medida más estricta de la cobertura, que las pruebas de sólo esas combinaciones de condiciones que puedan afectar la decisión, tome en cuenta que el 100% de la cobertura de condición múltiple implica el 100% de la cobertura de condición y decisión modificada.

Finalmente, hay una cobertura de bucle. Ésta no es una métrica formalizada de la cobertura estructural en el ISTQB nivel básico. Sin embargo, informalmente, podemos decir que hemos alcanzado el 100% de la cobertura del bucle cuando tengamos un conjunto de pruebas que obligan al programa a tomar todos los caminos del bucle cero, uno y muchas veces.

Nosotros mencionamos arriba que el 100% de la cobertura de rama significa que hemos alcanzado el 100% de la cobertura de sentencia. ¿Piensa usted de que el 100% de la cobertura de sentencia significa de que alcanzaremos el 100% de cobertura de rama? La respuesta es "no". Si usted ha escrito programas de computación antes, sabrá de que no cada sentencia if tiene una sentencia correspondiente else y que ningún constructo switch/case tiene un bloque por defecto de sentencias. En tales casos, el else implícito o la acción por defecto es "hacer nada", la cual representa una decisión tomada. Por su puesto, si haciendo nada es la cosa correcta para hacer, es

algo que también tenemos que probar.

¿Cómo se relaciona esto a las técnicas de diseño de pruebas basadas en la especificación que hemos abordado en la sección previa? Básicamente, la cobertura de rama o decisión corresponde a la cobertura de particiones de equivalencia. Las ramas dicen cómo el programa maneja las diferentes situaciones con fragmentos de código diferentes, que son creados y esas situaciones diferentes son las particiones de equivalencia.

Si probamos en los valores límite, probamos la implementación correcta de alguna de las condiciones. Sin embargo, porque el código podría implementar las combinaciones de condiciones, no significa que hemos logrado la cobertura de condición simple o condición múltiple. De este modo, la analogía entre las pruebas basadas en la especificación y la estructura se rompe en este punto.

```
1 #include <stdio.h>
2 main()
3 {
4     int i, n, f;
5     printf("n = ");
6     scanf("%d", &n);
7     if (n < 0) {
8         printf("Invalid: %d\n", n);
9         n = -1;
10    } else {
11        f = 1;
12        for (i = 1; i <= n; i++) {
13            f *= i;
14        }
15        printf("%d! = %d\n", n, f);
16    }
17    return n;
18 }
```

Figura 4.19: Ejemplo acerca de la Cobertura de Código

Veamos un ejemplo. El programa simple en C en esta página calcula un factorial. Un factorial es el producto de un entero dado y todos los enteros más pequeños, los cuales son mayor que cero. En otras palabras, el factorial de 3 es $3 \times 2 \times 1$ o 6. El factorial de 0 está definido como uno.

En este programa, la entrada es el número del cual se calcula el factorial. Está guardado en la variable *n*.

De este modo, para alcanzar la cobertura de sentencia, ¿Qué valores de pruebas necesitamos para *n*? Por favor calcúlelos ahora.

¿Listo? Bueno. Usted debería ver que si usted escoge un valor de *n* menor que 0 y otro valor de *n* mayor que 0, ejecutará cada sentencia al menos una vez.

Ahora, ¿Alcanzaría eso la cobertura de rama?

No, necesitamos también *n==0* para comprobar que el bucle no sea ejecutado.

Si probamos con *n* menor que 0, *n* igual a 0 y *n* mayor que cero, ¿Alcanzaría eso cobertura de condición? Sí, porque no tenemos condiciones compuestas.

Finalmente, ¿Qué hay de la cobertura de bucle? Para la cobertura de bucle del 100%, necesitamos de cubrir *n==1* y *n==el número máximo de veces en todo el bucle*.

De este modo, ¿Cómo podemos utilizar la cobertura de código para diseñar las pruebas? ¿Qué bueno es este material de las pruebas estructurales para nosotros como probadores?

En debates acerca de otras pruebas y con nuestra propia experiencia personal, nos hemos dado cuenta de que, por sí mismas, las técnicas de caja negra pueden dejar de cubrir hasta un 75% o más de las sentencias.

Ya sea si esta gran cantidad de código no probado representa un problema o no, depende de lo que no es cubierto, así como si los programadores probaron o no este código en las pruebas de unidad.

De este modo, como probadores independientes, podemos utilizar las herramientas de cobertura de código para instrumentalizar un programa. Esto nos permite monitorear la cobertura del código durante la ejecución. (Hablaremos más al respecto en el capítulo 5). Una vez que hayamos encontrado los vacíos en nuestra cobertura del código, decidiríamos de adicionar más casos de prueba para alcanzar niveles más altos de cobertura.

Al principio de este libro hablamos del análisis estático del código. Mencionamos que algunas herramientas de análisis estático pueden localizar segmentos del código altamente complejos. Veamos esa medida de complejidad, la cual tiene algunas implicaciones interesantes del diseño de pruebas.

La complejidad ciclomática de McCabe mide la complejidad del flujo de control. Específicamente, la métrica trabaja como sigue. Cada función en un programa, incluyendo la función principal, comienza con un contador de complejidad de uno. Contamos la complejidad función por función.

Cada vez que hay una rama o bucle en una función, el contador de complejidad aumenta en uno. En otras palabras, cuando Tom McCabe diseñó esta métrica, la diseñó para sostener su teoría de la complejidad de la programación, la cual dice que la programación es una tarea compleja a causa de las decisiones que deben ser creadas en el código para dirigir los flujos de control.

Por supuesto, la mejor manera de medir la complejidad ciclomática es utilizando una herramienta. La herramienta creará un grafo dirigido o diagrama de flujo desde el código. Los nodos (o las burbujas) representan los puntos de entrada, los puntos de salida y las decisiones. Las aristas (o las flechas) representan secuencias de cero o más sentencias sin ramas. Por supuesto, si tienen que crear el grafo y calcular la complejidad manualmente, usted puede, pero no es muy divertido para los programas del mundo real.

Porque no es muy divertido y las herramientas pueden ser costosas (sin embargo algunas son libres), ¿Por qué ocuparse con este concepto de la complejidad ciclomática? Bien, éste tiene algunas implicaciones útiles de las pruebas.

Por un lado, si McCabe tenía razón de que la complejidad de la tarea de programación se incrementa con el número de decisiones, los módulos que marcan alto en la complejidad ciclomática de McCabe, serán inherentemente defectuosos y propensos a la regresión. Hay diferentes opiniones acerca de si McCabe **tenía** razón.

Hemos leído un estudio de algunas personas en IBM, que examinaron una de sus aplicaciones. Para esta aplicación, ellos no encontraron correlación entre las varias métricas de complejidad para los varios módulos y las densidades de defectos para esos mismos módulos. Ahora, eso puede significar que la complejidad no influencia la defectuosidad. Sin embargo, eso podría también significar que las métricas de complejidad que ellos examinaron no se adecuan a la captura de todos los elementos de complejidad. ¿Tal vez lo que hace a un programa verdaderamente complejo —y así propenso a los defectos—es algo que la mayoría de las métricas de complejidad no miden?

Personalmente, tenemos alguna experiencia trabajando con código altamente complejo—acerca lo cual queremos decir las dos cosas, intuitivamente complejo así como también complejo en cuanto a la métrica ciclomática de McCabe. Hemos trabajado con tal código en ambos tipos de aplicaciones tanto comerciales como científicas. Hemos encontrado ciertamente situaciones, donde la complejidad del código no condujo a más defectos. De ese modo, diríamos que aunque no pudimos generar ninguna conclusiones finales acerca de que si McCabe tenía razón basados en la evidencia empírica hasta ahora, es todavía una regla de dedo útil para el análisis de los riesgos de calidad para afirmar que esperamos una alta probabilidad de defectos en esas partes del sistema las cuales son altamente complejas.

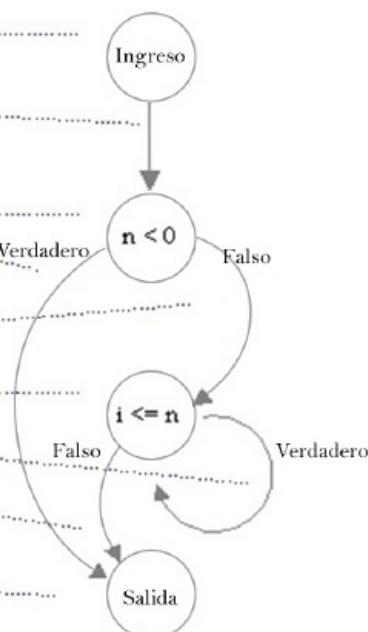
La otra implicación útil de esta métrica para las pruebas es cuando se evalúa un conjunto de pruebas de unidad para una función. Usted puede extender la técnica de las gráficas del flujo de control de McCabe para generar lo que él llama los caminos a través del grafo. El número de caminos básicos es igual al número de pruebas básicas requeridas para cubrir el grafo. Retornaremos a este concepto en breve, porque éste le ayudará a visualizar los caminos básicos y las pruebas básicas como lo hablamos.

Entonces examinemos más de cerca la complejidad ciclomática de McCabe...

Programa

```
main()
{
    int i, n, f;
    printf("n = ");
    scanf("%d", &n);
    if (n < 0) {
        printf("Invalid: %d\n", n);
        n = -1;
    } else {
        f = 1;
        for (i = 1; i <= n; i++) {
            f *= i;
        }
        printf("%d! = %d.\n", n, f);
    }
    return n;
}
```

Diagrama de Flujo



Complejidad Ciclomática

$$C = \#R + 1 = \\ 2 + 1 = 3$$

$$C = \#E - \#N + 2 = \\ 5 - 4 + 2 = 3$$

Definiciones
 C = Complejidad ciclomática
 R = Región encerrada
 E = Arista (flecha)
 N = Nodo (burbuja)

Figura 4.20: La Complejidad Ciclomática para el Factorial

Esta figura muestra el programa que calcula el factorial en el lado izquierdo. Las líneas punteadas del código al diagrama del flujo de McCabe en el centro de esta figura muestran cómo las secuencias sin ramas de cero o más sentencias llegan a ser las aristas (o las flechas), y cómo los constructos con las ramas y los bucles llegan a ser los nodos (o las burbujas).

En la parte derecha de la figura, usted puede observar dos métodos que calculan la métrica de la complejidad ciclomática de McCabe. El más simple es tal vez el cálculo de la “región cerrada”. Las dos regiones cerradas, representadas por R en la ecuación superior, se encuentran en el diagrama de abajo y mayormente en la izquierda del nodo “ $n < 0$ ” y abajo en la derecha del nodo “ $i \leq n$ ”.

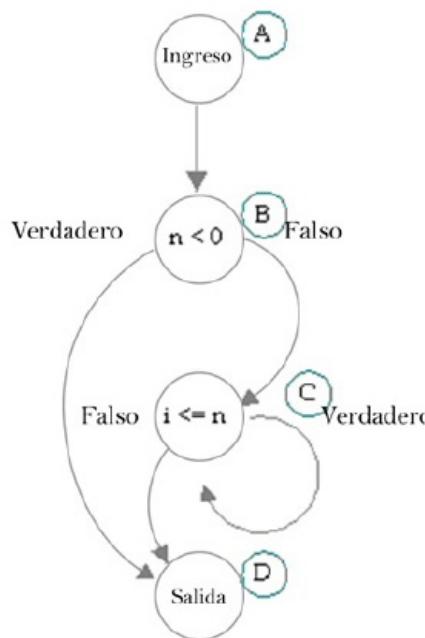
El otro método de cálculo involucra el conteo de las aristas (o flechas) y los nodos (o burbujas).

Ahora, esto es simplemente suficiente para un programa pequeño y simple como éste. Para funciones más grandes, dibujando el gráfico y haciendo el cálculo de éste es un verdadero lío. De este modo, una simple regla de dedo es: Cuente los constructos de rama y bucle y adicione 1. Las sentencias “if” y los constructos “for”, “while” y “do/while” cuentan como uno. Para los constructos switch/case, cada bloque case cuenta como uno. En los constructos “if” y “ladder if”, el “else” no se cuenta. Para los constructos switch/case, el bloque “por defecto” no se cuenta. Ésta es una regla heurística, pero parece que siempre funciona.

Programa

```
main()
{
    int i, n, f;
    printf("n = ");
    scanf("%d", &n);
    if (n < 0) {
        printf("Invalid: %d\n", n);
        n = -1;
    } else {
        f = 1;
        for (i = 1; i <= n; i++) {
            f *= i;
        }
        printf("%d! = %d.\n", n, f);
    }
    return n;
}
```

Diagrama de Flujos



Caminos Básicos

1. ABD
2. ABCD
3. ABCCD

Pruebas Básicas

Entrada	Esperado
1. -1	Inválido: -1
2. 0	0! = 1.
3. 1	1! = 1.

Figura 4.21: Caminos y Pruebas Básicas

Cuando introdujimos la complejidad de McCabe en páginas anteriores, mencionamos los caminos básicos y las pruebas básicas. Esta figura muestra los caminos básicos y las pruebas básicas en el lado derecho.

El número de caminos básicos es igual a la complejidad ciclomática. usted construye los caminos básicos comenzando con un camino cualquiera a través del diagrama, luego adicionando otro camino que cubre el mínimo número de aristas no cubiertas previamente, repitiendo este proceso hasta que todas las aristas hayan sido cubiertas por lo menos una vez.

Las pruebas básicas son las entradas y los resultados esperados asociados con cada camino básico. Usualmente, las pruebas básicas coincidirán con las pruebas necesarias para alcanzar la cobertura de rama. Esto tiene sentido, porque la complejidad se incrementa en cualquier momento en que más de una arista sale de un nodo en un diagrama de flujo de McCabe. En un diagrama de flujo de McCabe, una situación donde “más de una arista de un nodo” representa un constructo de ramas o de bucles.

¿Para qué sirve esta exquisita información? Bien, suponga que estuviese hablando con un programador acerca de sus pruebas de unidad. Usted pregunta cuántas entradas utilizaron ellos. Si ellos le dicen un número menor que la métrica de complejidad ciclomática de McCabe, para el código que ellos están probando, es una apuesta segura de que ellos no alcanzaron la cobertura de rama. Eso implica, como fue mencionado más antes, de que ellos no cubren las particiones de equivalencia.

4.4.1 Ejercicios

Ejercicio 1

Abajo encontrará un programa simple escrito en C que acepta una cadena con caracteres hexadecimales (entre otros caracteres no deseados). Éste ignora los otros caracteres y convierte los caracteres hexadecimales en una representación numérica.²⁸

Si prueba con las cadenas de entrada: "059", "ace" y "ACD" ¿Qué niveles de cobertura usted alcanzaría?

¿Qué cadenas de entrada podría usted añadir para alcanzar las coberturas de sentencia y rama?

¿Serían aquellas pruebas suficientes para probar este programa?

Argumente.

```
main()
{
    /* Convertir los dígitos hexadecimales en un número */
    int c;
    unsigned long int hexnum, nhex;
    hexnum = nhex = 0;
    while ((c = getchar()) != EOF) {
        switch (c) {
            case '0': case '1': case '2': case '3': case '4':
            case '5': case '6': case '7': case '8': case '9':
                /* Convertir un dígito decimal */
                nhex++;
                hexnum *= 0x10;
                hexnum += (c - '0');
                break;
            case 'a': case 'b': case 'c':
            case 'd': case 'e': case 'f':
                /* Convertir un dígito decimal de letra minúscula */
                nhex++;
                hexnum *= 0x10;
                hexnum += (c - 'a' + 0xa);
                break;
            case 'A': case 'B': case 'C':
            case 'D': case 'E': case 'F':
                /* Convertir un dígito decimal de letra mayúscula */
                nhex++;
                hexnum *= 0x10;
                hexnum += (c - 'A' + 0xA);
                break;
            default:
                /* Saltar cualquier carácter no hexadecimal */
                break;
        }
    }
    printf("Tengo %d dígitos hexadecimales: %x\n", nhex, hexnum);
    return 0;
}
```

Figura 4.22: Programa del Conversor Hexadecimal

Solución del Ejercicio 1

Las cadenas "059", "ace" y "ACD" no alcanzan ningún nivel específico de cobertura del cual hablamos en el libro. Necesitaríamos añadir "xyz" o alguna otra cadena que contenga dígitos no hexadecimales para alcanzar la cobertura de sentencia. Necesitaría probar la cadena nula para obtener la cobertura de rama (en cuanto a no ejecutar el bucle). Para tratar de ejecutar el bucle rigurosamente, también queríamos probar una cadena muy larga.

Mientras que no es necesario alcanzar ninguno de los niveles de cobertura estructural que abordamos, usted podría considerar cadenas cortas mezcladas. Por ejemplo, "6dpF" prueba cada uno de los cuatro bloques "case" en una sola prueba, y comprobar problemas que podrían ocurrir en esas situaciones.

4.5 Técnicas Basadas en la Experiencia

Objetivos del Aprendizaje

LO-4.5.1 Recordar las causas para escribir casos de pruebas basados en la intuición, experiencia y conocimiento acerca de los defectos comunes. (K1)

LO-4.5.2 Comparar las técnicas basadas en la experiencia con las técnicas de pruebas basadas en la experiencia. (K2)

En esta sección, Técnicas Basadas en la Experiencia, cubrirá los siguientes conceptos clave:

- Razones para escribir casos de prueba basados en la intuición, la experiencia y el conocimiento.
- Comparación de las técnicas basadas en la experiencia con las técnicas basadas en la especificación.

Comencemos con los conceptos básicos de las pruebas basadas en la experiencia.

En este punto, entenderá el diseño de pruebas de comportamiento o el diseño de pruebas basadas en la especificación. En las pruebas de comportamiento, ignoramos el funcionamiento interno del

sistema y nos enfocamos en cómo se supone que éste se comporte. También entenderá el diseño de pruebas estructurales, donde se observa el funcionamiento interno del sistema y como el sistema hace lo que hace.

Cuando experimentamos las pruebas basadas en la experiencia, ponemos a un lado la distinción entre pruebas estructurales y de comportamiento, y utilizamos todo lo que sabemos en nuestras pruebas. Las pruebas basadas en la experiencia se basan en la habilidad y la intuición del probador, su experiencia con aplicaciones similares y su experiencia con tecnologías similares.

Es común encontrar que en vez de ser prediseñadas, las pruebas basadas en la experiencia son a menudo creadas durante la ejecución de pruebas. En otras palabras, la estrategia es dinámica o reactiva. Hablaremos acerca de las estrategias de pruebas en el siguiente capítulo.

Al comienzo de este capítulo, hablamos acerca del análisis de los riesgos de calidad. De este modo usted entiende ahora cómo y cuándo utilizamos los riesgos de calidad para controlar nuestro esfuerzo de las pruebas y conseguimos la alineación del esfuerzo de las pruebas con la calidad del sistema y con el nivel de riesgo para esa calidad.

Las pruebas basadas en la experiencia tienen algunos problemas en cuanto a la alineación, a causa de la creación de las pruebas al vuelo durante la ejecución de las pruebas. Por ejemplo, ¿cómo podemos evitar la pérdida de todo nuestro tiempo en algunas áreas y de obtener la cobertura estrecha y limitada de las pruebas? ¿Cómo sabemos cuánto tiempo se debe pasar en las pruebas de cualquier área?

Un método común para la mitigación de estos problemas es utilizar una lista de cartas de pruebas ("test charters"), creada antes de la ejecución de las pruebas. El tiempo que se debe pasar en estas cartas de pruebas es frecuentemente "tiempo limitado". Es decir, los períodos cortos de las pruebas son concentrados en las condiciones específicas de las pruebas asociadas con cada carta.

Algunos ejemplos de las técnicas de pruebas basadas en la experiencia son la predicción de error, la cacería de defectos, "el rompimiento del software" basados en listas de comprobación o taxonomías de defectos, y las pruebas exploratorias.

Glosario del ISTQB

Pruebas exploratorias: Una técnica diseño de pruebas informal donde el probador controla activamente el diseño de las pruebas como aquellas pruebas son llevadas a cabo y utilizan información ganada mientras se prueba para diseñar pruebas nuevas y mejores.

Veamos más de cerca cómo esto funciona...

Con la mayoría de los métodos para las pruebas basadas en la experiencia —al menos los métodos profesionales y serios— usted no crea todas las pruebas durante la ejecución de pruebas. Usted tiene o desarrolla guías con anticipación.

Algunos ejemplos comunes de estas guías incluyen lo siguiente:

Una lista de comprobación es tal vez lo más simple de las guías, en la cual se incluirán descripciones cortas de las áreas de dos a cinco palabras, las características de calidad y los rasgos para probar. Si obtiene una lista genérica de comprobación u otra para otro sistema, necesitará adaptarla para su sistema para asegurar de que es efectiva en cuanto al descubrimiento de defectos y la cobertura de áreas clave.

Una taxonomía es una guía más complicada. Una taxonomía es una jerarquía de defectos clasificada por tipo, subtipo, y así sucesivamente. Luego intentará de encontrar defectos que son descubiertos en la taxonomía. Como se podría imaginar, las taxonomías de defectos más efectivas son aquellas creadas por sistemas similares al que está probando actualmente. Las taxonomías pueden ser efectivas para descubrir defectos, pero, como cualquier método de pruebas, están enfocadas completamente en los riesgos técnicos—la probabilidad de descubrir defectos, no influye en cuanto a la construcción de confianza; y la reducción de los riesgos de calidad es incompleta.

Una lista de ataques es otra guía, todavía más complicada. En la lista de ataques, no tiene solamente defectos metas que está buscando, sino también procedimientos de pruebas de alto nivel para descubrir esos defectos. Los ejemplos más famosos de esta técnica son los de James Whittaker's *How to Break Software and How to Break Software Security*. Sin embargo, en un ejemplo aún más antiguo—uno que precede al libro de Whittaker por más de cinco años—es el de Brian Marick's *Craft of Software Testing*. Nuevamente éste se enfoca en los riesgos técnicos —descubriendo tantos defectos como sea posible—y no en las demás contribuciones valiosas y potenciales que las pruebas pueden aportar al proyecto.

De esta misma manera como en el método de Marick, el concepto de Elisabeth Hendrickson ofrece un catálogo de problemas comunes y formas de descubrirlos.

Algo diferente es el concepto, desarrollado por Jon and James Bach, de utilizar un conjunto de cartas de pruebas. Una carta de prueba es una descripción corta de la prueba que debe ser realizada en cualquier lugar de dos a tres palabras hasta un par de frases.

Esto **puede** basarse en dónde esperamos descubrir los defectos, pero también puede basarse en las características, los aspectos y las áreas clave del sistema. Entonces, como tal, es más parecido

a un método de listas de comprobación. Con el conjunto apropiado de cartas de pruebas, esta técnica no solo puede descubrir los defectos, si no también ayuda a construir la confianza en el sistema. Al igual que en las listas de comprobación, las cartas de pruebas también permiten algún análisis de cobertura de pruebas en cuanto al mismo sistema en vez de solamente una lista de defectos, que quisimos buscar.

Como regla general, un equipo de pruebas profesional, dada la oportunidad, prepararía un primer bosquejo de estas guías con anticipación con algún nivel de detalle. Decimos “dada la oportunidad”, porque muchas veces somos metidos a situaciones de pruebas en el último minuto y no tenemos otra opción excepto la de ser netamente reactivos. Note, que si aún somos obligados a improvisar, utilizando una o más de estas guías pueden ayudar a poner alguna estructura alrededor de una situación que de otra manera sería caótica.

Lamentablemente, la realización de las pruebas sobre la marcha como la única actividad de las pruebas—típicamente referidas en forma despectiva como las pruebas informales—es bastante común. Sin ninguna estructura alrededor de esta actividad, las pruebas informales son usualmente inefectivas y realizadas de forma manual y aleatoria. Los probadores que realizan pruebas informales, se pasean en una gran nube infinita de las pruebas que ellos posiblemente podrían ejecutar, hasta que se les acaba el tiempo y luego tienen poco que mostrar acerca de las pruebas, pero un pequeño montón de informes de defectos. Esperamos que le sea clara la explicación acerca de las técnicas de pruebas basadas en la experiencia en este libro, y que el programa de estudios básico del ISTQB **no sea** un sello de la aprobación de esos métodos descuidados e ineficaces.

Al comienzo de este capítulo, explicamos lo esencial de la estrategia de pruebas analítica y basada en los riesgos. Identificamos los riesgos de calidad basados en el análisis de los requisitos, el diseño, y así sucesivamente, informado por las percepciones de los interesados clave.

Determinamos el nivel de riesgo, usualmente en términos de la probabilidad y el impacto. Luego diseñamos las pruebas para cubrir aquellos riesgos, que son lo suficientemente importantes para dedicar nuestra atención, cuanto más alto el nivel de riesgo, cuantas más pruebas debemos realizar. Cuanto más alto el nivel de riesgo asociado con una prueba, cuanto más temprano debemos ejecutar la prueba. Hemos mencionado también las oportunidades para la prevención de los defectos y la mitigación total de los riesgos del proyecto inherentes en esta estrategia **si** empezamos con anticipación el análisis de los riesgos de calidad y las actividades de la preparación de las pruebas subsiguientes, en paralelo con la especificación de los requisitos y la planificación del proyecto.

Ahora, hablaremos más ampliamente acerca de las estrategias de pruebas en el capítulo 5, como ya lo mencionamos antes, pero en este momento en el capítulo 4 nos encontramos confrontados con una estrategia de pruebas competitiva y una estrategia de pruebas dinámica o reactiva. En una estrategia de pruebas dinámica, pasamos menos tiempo en la preparación de las pruebas antes de la ejecución y nos enfocamos más en reaccionar al sistema que realmente recibimos. La mayoría de las estrategias dinámicas de pruebas se apoyan fuertemente en las técnicas basadas en la experiencia, simplemente porque hay tiempo insuficiente para llevar a cabo las técnicas más formales del diseño de pruebas—especialmente en la manera descrita en la sección 1 de este capítulo.

¿Entonces, cuáles son las ventajas y desventajas de estas estrategias?

Bien, como se podría imaginar que dado el enfoque acerca del descubrimiento de los defectos en las técnicas, estas estrategias son a menudo muy efectivas en encontrar defectos.

Además, debido a la especificación ligera de las pruebas, las pruebas tienden a variar de un probador a otro y de una ejecución de pruebas a otra. Así, éstas varían más cada vez que éstas son ejecutadas y resisten a la paradoja de pesticida explicada en el capítulo 1.

Porque la especificación es ligera, estas estrategias de pruebas son eficientes, si medimos la eficiencia en el costo por defecto encontrado.

Porque las técnicas son tan diferentes a las técnicas estructurales y de comportamiento, éstas sirven como una manera excelente para encontrar vacíos en las pruebas que preparamos utilizando las estrategias analíticas.

Finalmente, la gente encuentra estas estrategias divertidas y creativas, porque el descubrimiento de un defecto es a menudo uno de los aspectos más gratificantes, intelectualmente desafiantes y algunas veces graciosos del trabajo de las pruebas. Sin embargo, no todo es color de rosas. Como mencionamos muchas de las técnicas están enfocadas completamente en el descubrimiento de los defectos, entonces la cobertura de las áreas clave, las características y los rasgos de calidad son cuestionables. Aún los métodos basados en las listas de comprobación y basados en las cartas de pruebas (“test charters”) producen una cobertura con vacíos, porque las cartas y las listas de comprobación representan con frecuencia la noción de una persona acerca de lo que debe ser probado, no un consenso con todos los interesados.

Además, si las listas de comprobación o las cartas son desarrolladas o adaptadas, a la rápida o bajo otras presiones, la persona que crea o adapta estos documentos tenderá a olvidar cosas,

conduciendo a una cobertura aún menos completa.

Las estrategias analíticas de pruebas producen una lista sólida de las condiciones de pruebas, los ítems de los riesgos y similares, cuando son iniciadas en el comienzo del proceso de pruebas, éstas pueden ser utilizadas para estimar el esfuerzo necesario para preparar y ejecutar las pruebas. Aquellos productos del trabajo no estarán disponibles cuando se utilizan las estrategias dinámicas.

Además porque no comenzamos con anticipación, pero tendemos a enfocarnos en reaccionar al sistema que nos ha sido realmente presentado, hay poca oportunidad, si alguna, para la prevención de los defectos. Las pruebas de las pequeñas mejoras y los pequeños trabajos de desarrollo, pueden sobrevivir la experiencia del hallazgo de un número más grande de defectos que lo esperado durante las últimas etapas de las pruebas, pero para proyectos grandes eso es un desastre, por lo tanto la prevención de los defectos es clave.

En muchos casos - por ejemplo, el método de las pruebas exploratorias con las cartas de pruebas - hay una realización extensa de interrogatorios y debates para guiar el proceso, porque no hay un "plan de pruebas" de por sí. (Más acerca de esto en el capítulo 5). Todas estas sesiones de debates e interrogatorios funcionan para un equipo pequeño, idealmente colocado y en la misma zona horaria, pero ellos no crecerán a equipos grandes o distribuidos. Además, estas estrategias, mientras son útiles como un complemento para las estrategias analíticas, no tienden a funcionar bien en muchos proyectos si se apoyan exclusivamente sobre estas.

Finalmente, porque estas estrategias dinámicas se apoyan en las técnicas de pruebas basadas en la experiencia, éstas dependen de la disponibilidad de los probadores experimentados. Típicamente en la práctica, mucha gente involucrada en las pruebas, especialmente en la ejecución de las pruebas, no tienen la habilidad necesaria y la experiencia para realizar correctamente las pruebas basadas en la experiencia.

Personal	7 Técnicos	3 Ingenieros + 1 Jefe
Experiencia	> 10 años en total	< 20 años en total
Tipo de Prueba	Guiones Precisos	Exploratorias en base a Charters
Prueba Hrs./Día	42	6
Defectos Encontrados	928(78%)	261(22%)
Efectividad	22	44

Tabla 4.12: Caso de Estudio de Pruebas Exploratorias

Esta tabla muestra un caso de estudio del uso de las pruebas exploratorias como técnica complementaria en el proyecto que se apoyó principalmente en la estrategia de pruebas analítica basada en los riesgos. Teníamos un equipo de pruebas que consistía en siete técnicos de pruebas, tres ingenieros y un jefe de pruebas, Rex Black.

Los técnicos de pruebas tenían menos de 10 años de experiencia en pruebas en total de todos los siete y alguno no tenía ni un año de experiencia.

Los técnicos de pruebas eran básicamente contratados a bajo costo a quienes le dimos precisamente los guiones detallados de las pruebas los cuales ellos luego ejecutaron. Cada técnico de pruebas pasa normalmente 6 horas de 9 o 10 horas en el día en realidad ejecutando estos guiones de pruebas. Las reuniones, el correo electrónico, la actualización de los guiones de pruebas, la asistencia a los ingenieros de pruebas y otras funciones importantes consumieron el resto del tiempo. De este modo fueron cerca de 42 horas al día, que se utilizaron en la ejecución de los guiones de prueba.

Sin embargo entre los tres ingenieros de prueba y Rex Black, tenían más de 20 años en total de experiencia en pruebas, de hecho probablemente más de 30 años. De manera que no tendrían confianza en los guiones de pruebas y la capacidad de los técnicos de pruebas para ejecutar aquellos guiones, decidimos realizar alguna prueba exploratoria con cartas. Esto significó que nos pondríamos de acuerdo en probar diferentes áreas que pensamos, que merecieran nuestra atención. Sin embargo, debido a que estuvimos ocupados, pudimos dejar de lado solamente unas 6 horas al día en total entre los cuatro de nosotros para las pruebas exploratorias.

En otras palabras, las pruebas exploratorias representaron un 13% del esfuerzo diario de las pruebas. Pero, como puede ver en la tabla, mientras que las pruebas basadas en guiones encontraron 928 defectos, un 78%, las pruebas exploratorias, basadas en el esfuerzo, encontraron desproporcionadamente un numero alto, 261, o alrededor del 22%. Si divide el número total de defectos por las horas por día del esfuerzo de pruebas, nos da una métrica de efectividad de 22

para las pruebas basadas en guiones y un 44 para las pruebas exploratorias.

De este modo, este caso de estudio nos demuestra que las pruebas exploratorias son alrededor de 2 veces más efectivas en descubrir defectos en base a hora por hora. Si fuéramos a calcular la eficiencia e incluir el tiempo dedicado para la preparación de las pruebas, las pruebas exploratorias serían significativamente más eficientes. Después de todo, las pruebas basadas en guiones necesitaron un esfuerzo extenso para ser creadas. Este caso de estudio apoya el uso de pruebas exploratorias en un proyecto.

Sin embargo, hay algunas cosas de tomar en cuenta. ¿Podríamos haber empleado los técnicos de pruebas para realizar las pruebas exploratorias y descubrir aún más defectos? Probablemente no, porque el nivel relativo de experiencia es importante.

Además, como el cliché de gestión dice que lo que es medido es realizado y lo que no es medido no es realizado. Así el problema con esta figura es que estamos midiendo solamente los defectos. ¿Cuál es el valor de las pruebas más allá de encontrar los defectos? Bien, para uno es la construcción de confianza y para otros la mitigación de los riesgos. Porque las pruebas exploratorias tuvieron una medida de cobertura pequeña y ninguna trazabilidad hacia atrás a los riesgos, no hay mucha cobertura demostrable o mitigación de los riesgos.

Finalmente, note que la regresión fue un mayor riesgo para este proyecto. Tuvimos que ser capaces de realizar pruebas precisas de regresión, como es el caso en muchos proyectos. Así, el valor de la reutilización de los guiones fue de valor significativo.

4.5.1 Ejercicios

Ejercicio 1

Considere la utilización de las pruebas reactivas como opuestas a las pruebas prediseñadas. En la tabla de abajo, ponga un "+" en la columna donde el factor o interés motiva hacia la utilización del método y un "-" en la columna donde el factor o interés desmotiva la utilización del método.

Factores o intereses importantes	Reactivas	Prediseñadas
Realizar pruebas de regresión rigurosas.		
Maximizar la eficiencia del hallazgo de los defectos.		
Emplear probadores con poca experiencia.		
Automatizar la mitad de los casos de prueba.		
Entregar estimaciones precisas y exactas.		
Probar sin tiempo de preparación de las pruebas.		
Probar una Interfaz de Usuario ("UI") que cambia rápidamente.		
Emplear un trabajo distribuido de las pruebas.		

Tabla 4.13: Ejercicio-¿Pruebas Reactivas o Prediseñadas?

Solución del Ejercicio 1

Factores o intereses importantes	Reactivas	Prediseñadas
Realizar pruebas de regresión rigurosas.	-	+
Maximizar la eficiencia del hallazgo de los defectos.	+	-
Emplear probadores con poca experiencia.	-	+
Automatizar la mitad de los casos de prueba.	-	+

Entregar estimaciones precisas y exactas.	-	+
Probar sin tiempo de preparación de pruebas.	+	-
Probar una Interfaz de Usuario ("UI") que cambia rápidamente.	+	-
Emplear un trabajo distribuido de las pruebas.	-	+

Tabla 4.14: Solución-Pruebas Reactivas o Prediseñadas

Los métodos reactivos de pruebas como los ataques de Whittaker o las técnicas exploratorias de Kaner/Bach/Bolton tienden a tener un gran número de defectos identificados por las horas totales de las pruebas. Porque existe una mínima documentación, usted puede empezar a probar inmediatamente, lo cual es útil cuando usted no está involucrado en el proyecto desde el principio. (Suponemos que es como hacer de la necesidad una virtud, porque la participación tardía de los probadores es una de las peores prácticas de ingeniería de software). También debido a la mínima documentación, los cambios en la interfaz de usuario e incluso en la funcionalidad no incurren en grandes costos del mantenimiento de las pruebas.

Sin embargo, cuando usted necesita realizar pruebas de regresión rigurosas— especialmente si las repeticiones de las pruebas deben cubrir exactamente las mismas condiciones como en la ronda inicial de las pruebas— entonces, las pruebas prediseñadas y bien documentadas son obligatorias. Cuando el equipo de pruebas es menos experimentado, las pruebas prediseñadas pueden servir de guía útil, y esas pruebas también pueden ayudar a un equipo distribuido de pruebas a dividirse el trabajo de las pruebas. Éstas también pueden ser guías útiles para los automatizadores de las pruebas, especialmente si están diseñadas con un ojo hacia la automatización final. Finalmente, los casos de prueba escritos, junto con el seguimiento cuidadoso de la duración y el esfuerzo involucrados en los trabajos previos de las pruebas, son útiles cuando se estima el esfuerzo de las pruebas.

4.6 Selección de las Técnicas de Pruebas

Objetivos del Aprendizaje

LO-4.6.1 Clasificar las técnicas de diseño de pruebas según su aptitud para un contexto dado, para la base de pruebas, para los modelos y características de software respectivos. (K2).

Esta sección, Selección de las Técnicas de Pruebas, cubrirá los siguientes conceptos clave, los cuales son, los factores que influencian la selección de las técnicas apropiadas del diseño de pruebas.

¿Entonces cuáles son los factores que influencian la selección de las técnicas del diseño de pruebas?

Tipo de sistema: Es una buena idea de utilizar las tablas de decisión para probar sistemas que procesan transacciones similares. Sistemas que deben comportarse diferente dependiendo de lo que ha ocurrido hasta el momento son más susceptibles a las pruebas basadas en los estados.

Estándares regulatorios: Por Ejemplo, las Regulaciones de la Administración de la Aviación Federal de los Estados Unidos requiere que los probadores usen técnicas estructuradas de pruebas para asegurar un cierto nivel de cobertura del código en el software de aviónica de seguridad crítica.

Requisitos del cliente o el contrato: El cliente o el contrato podría requerir que ciertas técnicas sean utilizadas, directamente o indirectamente.

Nivel y tipo de riesgo: Deberíamos probar ciertos sistemas, como los de seguridad crítica, en los niveles más altos posibles de seguridad. Deberíamos utilizar todas las técnicas de pruebas aplicables.

Objetivos de las pruebas: Cuanto mayor sea el nivel de detección de defectos que queremos alcanzar, más diversas son las técnicas que debemos utilizar. Si el objetivo principal es encontrar muchos defectos como sea posible en un corto tiempo, entonces confiaremos probablemente y fuertemente en las pruebas basadas en la experiencia.

Documentación disponible: Si tenemos especificaciones, podemos utilizar las técnicas basadas en la especificación fácilmente. Sin esas especificaciones, tenemos que confiar más en nuestra propia experiencia.

Conocimiento de los probadores: Los probadores con conocimiento pueden confiar en su conocimiento y su experiencia, y realizar más pruebas basadas en la experiencia.

Plazo y presupuesto: Si tenemos tiempo para preparar las pruebas, podemos hacerlo así. Si tenemos dinero para herramientas de cobertura de código, entonces podemos determinar la

cobertura estructural.

Ciclo de vida del desarrollo: Como lo hablamos al principio, ciertos modelos del ciclo de vida, como los iterativos, tienden a sufrir de los riesgos de alta regresión. Eso significa que tenemos que buscar pruebas reutilizables.

Experiencias previas acerca de los tipos de defectos encontrados: Si tenemos una buena idea de dónde buscar los defectos, las técnicas basadas en la experiencia que se basan en esa experiencia funcionarán bien para nosotros.

Por supuesto, hay otros factores. Es importante, que cuando piense acerca de cómo probar, que mantengan la mente comprometida con las necesidades del proyecto y vincule sus pruebas a esas necesidades.

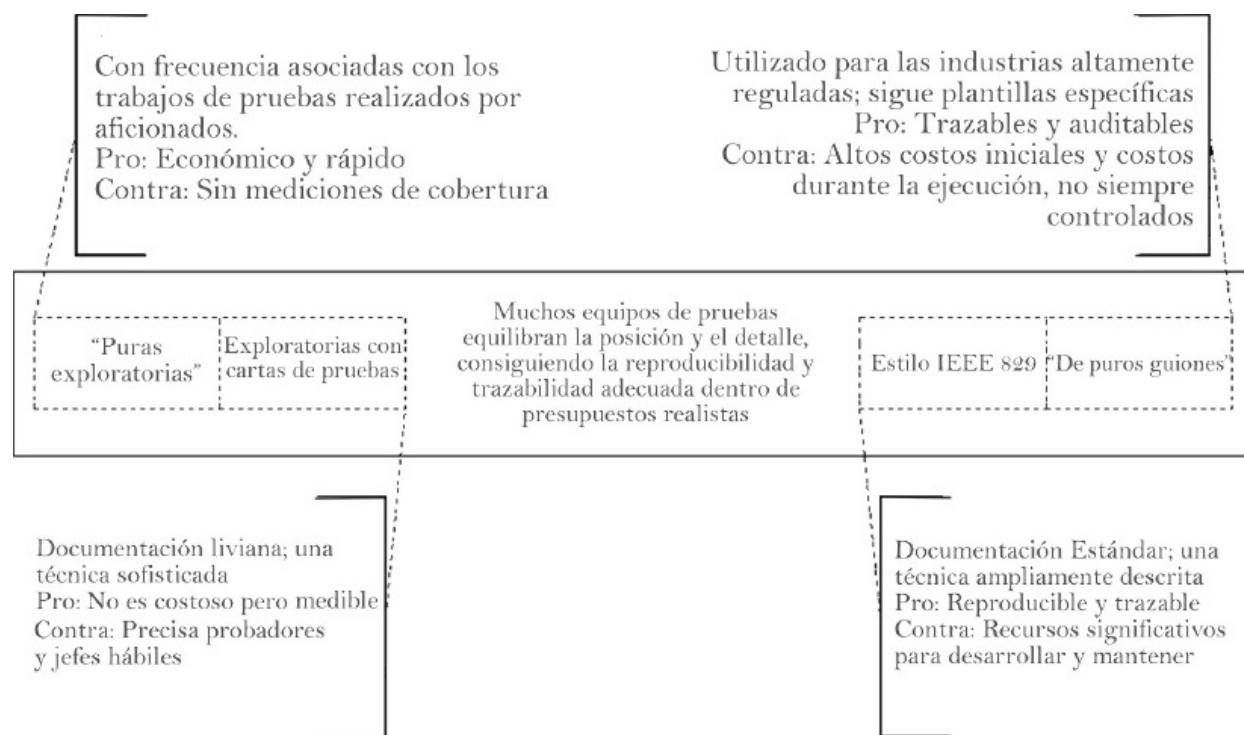


Figura 4.23: El Espectro Dinámico o Prediseñado

En la sección anterior acerca de las técnicas basadas en la experiencia y las estrategias dinámicas de pruebas, podría haber tenido la idea de que estas técnicas son completamente diferentes de otros tipos de técnicas y estrategias de pruebas, especialmente aquellas que producen guiones prediseñados.

Sin embargo, la realidad es que hay más un espectro que una dicotomía. Cuanto más precisas y detalladas las pruebas, cuanto más aprovechamos de los beneficios de la trazabilidad, la reproducibilidad y aún la auditabilidad. Sin embargo pagamos el precio en cuanto al costo del desarrollo y el mantenimiento, así como también la incursión del riesgo de que los probadores pudieran desviarse de todas maneras de los guiones, así privándonos de los beneficios que buscamos.

Cuanta menos especificación alrededor de las pruebas, cuanto más aprovechamos los beneficios como la disminución de los costos de las pruebas, la flexibilidad mejorada y la respuesta a las necesidades de cambio del proyecto. Sin embargo, pagamos el precio en cuanto a la cobertura conocida, la reproducibilidad y algo similar.

La realidad es, una vez que usted separe varias personas quienes apoyan una técnica o estrategia sobre otra, los equipos inteligentes de pruebas equilibran la el tamaño de la documentación basada en las necesidades del proyecto. A menudo tenemos que vivir con las restricciones como el programa y el presupuesto y a menudo necesitamos a pruebas escritas y reproducibles. De este modo, tenemos que alcanzar un equilibrio.

Una vez estábamos conversando con un jefe de pruebas acerca del grado de detalle que él quería en sus casos de prueba. El dijo, "Bien, si puedo dar el caso de prueba a un gorila afeitado y esperaría que el gorila lo ejecute, ése es un caso de prueba bien escrito".

Hicimos la pregunta obvia. "¿Cuántos gorilas afeitados tiene en su equipo de pruebas?"
"Ninguno".

"¿Entonces por qué está haciendo esto?"

El no tenía una respuesta. Alguien le había dicho una vez que el estándar del gorila afeitado era una regla difícil y rápida, y él tomó esto como la sabiduría revelada.

Es importante que tome en cuenta cómo alcanzar el equilibrio correcto de esta pregunta acerca del detalle y la precisión del caso de prueba. La manera de alcanzar este equilibrio es de considerar los intercambios en el espectro de la documentación de las pruebas. Las pruebas precisas le permiten utilizar probadores con menos habilidad, pero esas pruebas no son muy flexibles. Las pruebas imprecisas pueden cubrir más condiciones—porque ellas son más rápidas de escribir—pero esas pruebas no son muy reproducibles, especialmente a través de múltiples probadores. Las pruebas precisas le proporcionarán los criterios de pruebas transparentes a usted y cualquiera que las lea, pero esas pruebas son difíciles y costosas para mantener. Las pruebas imprecisas son rápidas de escribir, pero la cobertura que ellas proporcionan, puede ser difícil de definir y medir.

La mayoría de los equipos de prueba no tienen guías para el grado correcto de precisión, lo cual significa que ellos típicamente documentan en exceso o documentan muy poco. Un buen comienzo es medir el número de palabras de la documentación en un ejemplo aleatorio de 10, 50 o mejor todavía 100 de sus casos de prueba y procedimientos de prueba. Ahora, por cada caso de prueba, divida el número de palabras entre el número de horas o minutos que toma la ejecución de esa prueba. Cuanto más grande el número, cuanto más precisa la documentación que está proporcionando a sus probadores.

A menudo, cuando realizamos este ejercicio con los clientes encontramos diferencias de orden de magnitud entre los probadores acerca de esta métrica. Adicionalmente, no hay una guía de gestión que diga a quién se debe proporcionar cuánto detalle. Ése es un problema para rectificar si es que existe en su organización.

Glosario del ISTQB

Técnica de diseño de pruebas de caja negra: Procedimiento para derivar y/o seleccionar casos de prueba basados en un análisis de la especificación, ya sea funcional o no funcional, de un componente o sistema sin referencia a su estructura interna.

Técnica de diseño de pruebas basada en la experiencia: Este término no está definido en el Glosario ISTQB. La política del ISTQB es que los términos no definidos en el glosario no pueden entrar en el examen.

Técnica de diseño de pruebas basada en la especificación: Véase técnica de diseño de pruebas de caja negra.

Técnica de diseño de pruebas basada en la estructura: Véase técnica de diseño de pruebas de caja blanca.

Técnica de diseño de pruebas de caja blanca: Procedimiento para derivar y/o seleccionar casos de prueba basados en un análisis de la estructura interna de un componente o sistema.

4.6.1 Ejercicios

Ejercicio 1

Hacer una lista de los factores abordados en esta sección que afectarían a la selección de las técnicas de pruebas y el alcance de la documentación de Omninet.

Argumente.

Solución del Ejercicio 1

En cuanto a los factores que afectarían a las técnicas de pruebas y el alcance de la documentación, incluiríamos los siguientes:

- Tipo de sistema (p.ej. estado versus decisión).
- Estándares regulatorios (p.ej. accesibilidad).
- Requisitos del cliente.
- Capacidad del conocimiento de los probadores (Alcance de la documentación).
- Riesgos (p.ej. cuál nivel de cobertura de código podría ser apropiado).
- Objetivos de las pruebas (especialmente examinando los límites los que más nos preocupan)
- Documentación disponible (porque podemos utilizar como bases de pruebas el Documento de los Requisitos de Marketing y el Documento de los Requisitos del Sistema Omninet).
- Tiempo y presupuesto (afectando el grado general de pruebas).

En cuanto a los factores que no afectarían a las técnicas de pruebas y el alcance de la documentación, incluiríamos los siguientes:

- Requisitos contractuales. Ciclo de vida del desarrollo.
- Experiencias previas (este factor aplicaría en proyectos posteriores, pero no en éste, el cual es versión 1.0).

Preguntas de Examen de Muestra y Simulación

Para finalizar cada capítulo, usted puede tratar de resolver una o más preguntas de examen de muestra para reforzar su conocimiento y comprensión del material y prepararse para el examen del Probador ISTQB Nivel Básico.

Sección 4.1 El proceso de desarrollo de pruebas (K3)

Estándares

- [IEEE 829] Estándar IEEE 829™ (1998/2005) El estándar IEEE para la Documentación de las Pruebas de Software]

Términos

Especificación de caso de prueba, diseño de prueba, cronograma de la ejecución de pruebas, especificación de procedimiento de prueba, guión de pruebas y trazabilidad.

#	Pregunta	K
87.	<p>Objetivo del aprendizaje: LO-4.1.1</p> <p>Considere los siguientes productos del trabajo de las pruebas y sus propósitos:</p> <ol style="list-style-type: none">I. La especificación del diseño de las pruebas.II. La especificación del caso de prueba.III. La especificación del procedimiento de prueba.IV. Especifica la secuencia de las acciones de las pruebas.V. Especifica los resultados esperados de las pruebas.VI. Especifica las condiciones de las pruebas. <p>¿Cuál de las siguientes afirmaciones emparejan correctamente el producto del trabajo con su propósito?</p> <ol style="list-style-type: none">A. I va con IV; II va con V; III va con VI.B. I va con VI y V; II o III puede ir con IV.C. I va con VI; II va con V; III va con IV.D. I va con VI; II va con IV; III va con V.	2
88.	<p>Objetivo del aprendizaje: LO-4.1.2</p> <p>Usted está probando un sistema de comercio electrónico.</p> <ol style="list-style-type: none">I. Un área de interés es la habilidad de cargar una compra a una tarjeta de crédito compatible durante el proceso del pago.II. Usted obtiene una lista de todas las tarjetas de crédito compatibles.III. Usted documenta las acciones que deben ser tomadas cuando se está probando el proceso del pago. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ol style="list-style-type: none">A. I, se ocupa de las condiciones de prueba; II es parte del caso de prueba; III es parte del procedimiento de prueba.B. III se ocupa de las condiciones de prueba; I es parte del caso de prueba; II es parte del procedimiento de prueba.C. I, II, y III pertenecen a una especificación del diseño de las pruebas.D. I, II, y III pertenecen al plan de prueba.	2
89.	<p>Objetivo del aprendizaje: LO-4.1.3</p> <p>Considere el siguiente fragmento de un caso de prueba. Los siguientes son entradas inválidas para el campo Cantidad:</p> <p>Cero ("0") Un ítem más que la máxima cantidad del pedido. Números no enteros (P.ej. "1.5"). Números negativos (P.ej. "-1"). Letras. Puntuación. Entrada nula (nada).</p> <p>Cadenas de texto muy largas que consisten de solo dígitos.</p> <p>Consultar la guía del usuario y la ayuda en línea para determinar la cantidad máxima de pedidos, así como también el mensaje de error apropiado que se debería mostrar con estas condiciones. Verificar los mensajes de error apropiados y la posibilidad de corregir el campo y continuar una vez que un valor válido sea ingresado. ¿Cuál de los siguientes elementos de un caso de prueba es encontrado en este fragmento del caso de prueba?</p> <ol style="list-style-type: none">A. Los resultados esperados.	2

	B. La clara trazabilidad a los requisitos. C. Las precondiciones de la ejecución. D. Las post condiciones de la ejecución.	
90.	<p>Objetivo del aprendizaje: LO-4.1.4</p> <p>Un analista de negocios desempeñando un rol de probador escribió el siguiente fragmento de un procedimiento de prueba.</p> <ol style="list-style-type: none"> 1. Crear una nueva cuenta de usuario. Tratar de llevar a cabo acciones inválidas en esa cuenta. Verificar el rechazo de esas acciones. 3. Llevar a cabo una secuencia de transacciones válidas para la cuenta. Asegurar que los datos financieros estén en equilibrio. 4. Eliminar la cuenta. Asegurar que la información de la cuenta sea archivada en vez de eliminada. <p>Considere las siguientes suposiciones, que este procedimiento de prueba podría hacer acerca de las personas quienes ejecutarán la prueba.</p> <ol style="list-style-type: none"> I. Ellas comprenden la manera cómo el software funciona; es decir, cómo lograr tareas particulares con éste. II. Ellas comprenden el problema del negocio que el software soluciona y serían capaces de diferenciar el comportamiento correcto del incorrecto. III. Ellas comprenden la estructura interna del sistema. IV. Ellas estuvieron involucradas en la escritura de la especificación de los requisitos. V. Ellas tienen permiso para crear y eliminar cuentas y para inspeccionar los datos archivados. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ol style="list-style-type: none"> A. Todas son suposiciones necesarias acerca de la gente quien ejecutará la prueba. B. I, II y III son suposiciones necesarias acerca de la gente quien ejecutará la prueba, pero IV y V son innecesarias. C. Todas son suposiciones innecesarias acerca de la gente quien ejecutará la prueba, porque es una prueba detallada. D. I, II, y V son suposiciones necesarias acerca de la gente quien ejecutará las pruebas, pero III y IV son innecesarias. 	3
91.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es trazabilidad?</p> <ol style="list-style-type: none"> A. La capacidad del producto de software para hacer posible que el software modificado sea probado. B. La capacidad de identificar los ítems relacionados en la documentación y el software, así como los requisitos con las pruebas asociadas. C. Un método para las pruebas de integración, donde el componente es probado primero en la parte superior de la jerarquía de componentes. D. El grado en el cual un requisito es formulado en términos que permiten el establecimiento de diseños de pruebas. 	1
92.	<p>Objetivo del aprendizaje: Estándar IEEE 829.</p> <p>¿Cuál de las siguientes es una sección importante en la plantilla de especificación de diseño del IEEE 829?</p> <ol style="list-style-type: none"> A. Refinamientos del método. B. Especificaciones de salida. C. Especificaciones de entrada. D. Características que no deben ser probadas. 	1
93.	<p>Objetivo del aprendizaje: Estándar IEEE 829.</p> <p>¿Cuál de las siguientes es una sección importante en la plantilla de especificación de casos de prueba del IEEE 829?</p> <ol style="list-style-type: none"> A. Pasos del procedimiento. B. Necesidades del entorno. C. Criterios de paso/falla de las características. D. Riesgos y contingencias. 	1
94.	<p>Objetivo del aprendizaje: Estándar IEEE 829.</p> <p>¿Cuál de las siguientes es una sección importante en la plantilla de procedimiento de los casos de prueba del IEEE 829?</p> <ol style="list-style-type: none"> A. Ítems de prueba. B. Propósito. C. Identificación de la prueba. 	1

- D. Criterios de suspensión/reanudación.

Sección 4.2 Categorías de las técnicas de diseño de prueba (K2)

Términos

Técnica de diseño de pruebas de caja negra, técnica de diseño de pruebas basada en la experiencia, técnica de diseño de pruebas basada en la especificación, técnica de diseño de pruebas basada en la estructura y técnica de diseño de pruebas de caja blanca.

#	Pregunta	K
95.	<p>Objetivo del aprendizaje: LO-4.2.1 ¿Cuál de las siguientes es una razón por qué los métodos de pruebas basados en la estructura son útiles?</p> <ul style="list-style-type: none"> A. Los métodos de pruebas basados en la estructura encuentran más defectos que fallas. B. Los métodos de pruebas basados en la estructura no necesitan herramientas. C. El grado de la cobertura del software puede ser medido para los casos de prueba existentes, y los casos de prueba adicionales pueden ser derivados sistemáticamente para incrementar la cobertura. D. Los modelos de la especificación son utilizados para derivar sistemáticamente los casos de prueba. 	1
96.	<p>Objetivo del aprendizaje: LO-4.2.2 Usted está desarrollando pruebas por medio del análisis de un conjunto de casos de uso preparados por el analista de negocios. ¿Cuál tipo de método de pruebas está utilizando?</p> <ul style="list-style-type: none"> A. Basado en la estructura. B. Basado en la experiencia. C. Informal ("Ad hoc"). D. Basado en la especificación. 	2
97.	<p>Objetivo del aprendizaje: término. ¿Qué es una técnica de diseño de pruebas de caja negra?</p> <ul style="list-style-type: none"> A. Un procedimiento para derivar y/o seleccionar casos de prueba basados en un análisis de la especificación. B. Un procedimiento para derivar y/o seleccionar casos de prueba basados en un análisis de la estructura interna. C. Un procedimiento informal donde el probador controla activamente el diseño de las pruebas a medida que éstas son realizadas. D. Un procedimiento para probar todas las combinaciones de los valores de entrada y las precondiciones. 	1

Sección 4.3 Técnicas basadas en la especificación o de caja negra (K3)

Términos

Análisis de valores límite, pruebas de tablas de decisión, particionamiento de equivalencia, pruebas de estados de transición y pruebas de casos de uso.

#	Pregunta	K
98.	<p>Objetivo del aprendizaje: LO-4.3.1 Un campo acepta una entrada de un entero entre 1 y 99 representando la cantidad de un artículo que debe ser comprado. Considere los siguientes números:</p> <ul style="list-style-type: none"> I. 0 II. -7 III. 1 IV. 52 V. 99 VI. 100 VII. 129 <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. II, IV y VII son valores límite, mientras que I, III, V y VI son miembros de las particiones de equivalencia <i>inválido-demasiado bajo</i>, <i>válido</i> e <i>inválido</i>. 	3

	<p><i>demasiado alto.</i></p> <ul style="list-style-type: none"> B. I, III, V y VI son valores límite, mientras que II, IV y VII son miembros de las particiones de equivalencia <i>inválido-demasiado bajo</i>, <i>válido</i>, e <i>inválido-demasiado alto</i>. C. Todos los siete valores son valores límite y miembros de una de las tres particiones de equivalencia. D. Sólo I, III, V y VI son miembros de una de las tres particiones de equivalencia. 																			
99.	<p>Objetivo del aprendizaje: LO-4.3.1</p> <p>Usted está probando un sistema de comercio electrónico. Para pagar su compra, el sistema acepta cuatro tipos diferentes de tarjetas de crédito, cada una de las cuales tiene sus propias reglas para números de tarjetas válidos e inválidos. Una parte de la tabla de decisiones para el manejo de los pedidos es la siguiente:</p> <table border="1"> <thead> <tr> <th>Condición</th> <th>Sí</th> <th>No</th> </tr> </thead> <tbody> <tr> <td>Número de tarjeta inválido</td> <td>No</td> <td></td> </tr> <tr> <td>Compra aprobada</td> <td>No</td> <td>Sí</td> </tr> <tr> <td>Acción</td> <td></td> <td></td> </tr> <tr> <td>Mensaje de rechazo</td> <td>Sí</td> <td>No</td> </tr> <tr> <td>Procesar pago</td> <td>No</td> <td>Sí</td> </tr> </tbody> </table> <p>Observe que la combinación de las condiciones del <i>número de tarjeta inválido</i> y la <i>compra aprobada</i> no puede ocurrir. Asuma que quiere probar de tal forma que cubra completamente las combinaciones de las particiones de equivalencia para los tipos de tarjetas y las reglas mostradas en esta parte de la tabla de decisiones, ¿Cuántas pruebas necesita usted (sólo para el proceso descrito en esta parte de la tabla de decisiones)?</p> <ul style="list-style-type: none"> A. 12 B. 8 C. 9 D. 3 	Condición	Sí	No	Número de tarjeta inválido	No		Compra aprobada	No	Sí	Acción			Mensaje de rechazo	Sí	No	Procesar pago	No	Sí	3
Condición	Sí	No																		
Número de tarjeta inválido	No																			
Compra aprobada	No	Sí																		
Acción																				
Mensaje de rechazo	Sí	No																		
Procesar pago	No	Sí																		
100.	<p>Objetivo del aprendizaje: LO-4.3.1</p> <p>Un cajero automático tiene el siguiente diagrama de transición de estados para el control de retiros:</p> <p>Asuma que quiere desarrollar el mínimo número de pruebas para cubrir cada transición en el diagrama de transición de estados. Asuma además que cada prueba <i>debe empezar</i> en el estado inicial, <i>Esperando por el cliente</i>, y <i>debe terminar</i> cualquier momento que retorne al estado inicial, <i>Esperando por el cliente</i>. ¿Cuántas pruebas necesita usted?</p> <ul style="list-style-type: none"> A. 1 B. 3 C. 5 D. Infinitas 	3																		
101.	<p>Objetivo del Aprendizaje: LO-4.3.2</p> <p>¿Cuál es la regla de la cobertura mínima para una tabla de decisión?</p> <ul style="list-style-type: none"> A. Que cubra cada combinación de las condiciones. B. Que cubra cada acción posible. C. Que cubra cada columna en la tabla. D. Que cubra cada condición verdadera y falsa. 	2																		
102.	<p>Objetivo del Aprendizaje: LO-4.3.3</p> <p>¿Cuál es un beneficio típico de las pruebas de caso de uso?</p> <ul style="list-style-type: none"> A. Encontrar defectos en cada partición de equivalencia. B. Encontrar defectos en cada valor límite. 	2																		

	C. Encontrar defectos en las posibles transiciones de estado. D. Encontrar defectos en los flujos del proceso durante su utilización en el mundo real.	
103.	Objetivo del aprendizaje: término ¿Qué es un valor límite? A. Un valor de entrada o valor de salida el cual está en el borde de una partición de equivalencia o en la más pequeña distancia incremental en ambos lados del borde. B. Una tabla que muestra las combinaciones de las entradas y/o los estímulos con sus salidas asociadas y/o acciones. C. Un diagrama que describe los estados que un componente o sistema puede asumir, y muestra los eventos o las circunstancias que causan y/o resultan de un cambio de un estado a otro. D. Una parte de un dominio de entrada o salida para el cual se supone que el comportamiento de un componente o sistema sea el mismo.	1

Sección 4.4 Técnicas basadas en la estructura de caja blanca (K3)

Términos

Cobertura de código, cobertura de decisión, cobertura de sentencias y pruebas basadas en la estructura.

#	Pregunta	K
104.	Objetivo del aprendizaje: LO-4.4.1 Usted está a cargo de las pruebas de componentes de las nuevas funciones que han sido añadidas a un sistema. Usted quiere aumentar sus pruebas basadas en la especificación para estas funciones con las pruebas basadas en la estructura para los componentes a medida que ellos son escritos. ¿Cuál de las siguientes formas de medición de la cobertura estructural es la más probable para aplicar en este nivel? A. La cobertura de sentencia y decisión. B. La cobertura de árboles de llamadas. C. La estructura del menú. D. La cobertura estructural no es utilizada en el nivel de componente.	1
105.	Objetivo del aprendizaje: LO-4.4.2 Mientras se está probando una aplicación financiera, usted ha alcanzado la cobertura completa de las pruebas basadas en la especificación basado en las reglas para la cobertura de las particiones de equivalencia, los valores límite, las tablas de decisión, los diagramas de transición de estados y los casos de uso. Usted utiliza una herramienta de cobertura de código para medir la cobertura del código de las pruebas basadas en la especificación. Usted encuentra que el 30% de las decisiones están cubiertas por nuestras pruebas existentes. ¿Basado en la cobertura del código, qué decisión tomaría? A. Usted agrega pruebas adicionales para alcanzar el 100% de la cobertura de sentencia, porque eso produciría el 100% de la cobertura de decisión. B. Usted decide que la herramienta ha calculado incorrectamente la cobertura de decisión e informa acerca de un defecto al proveedor de la herramienta. C. Usted decide que el código no cubierto es inalcanzable y pide a los desarrolladores que lo eliminén. D. Usted agrega pruebas adicionales para cubrir las decisiones importantes que no fueron cubiertas por sus pruebas basadas en la especificación.	2
106.	Objetivo del aprendizaje: LO-4.4.3 Considere la siguiente función: int factorial(int n) int factorial(int n) /* Calcula el factorial utilizando recursión. */ /* Un factorial de un número es el producto */ /* del número mismo por el factorial */ /* del número menos 1, es decir, */ /* n! = n*(n-1)! */ /* Los factoriales de 0 y 1 son ambos 1. */ { int f=1; if (n < 0) { fprintf(stderr, "factorial: Argumento negativo.\n"); } else if ((n == 0) (n == 1)) { f=1; } else { f=n*factorial(--n); } return(f); }	3

Suponga que tiene un arnés de pruebas que le permitirá presentar valores de prueba a la función factorial (las entradas) y para comprobar los valores que devuelve (las salidas). ¿Cuál del siguiente conjunto de casos de prueba dan las correctas especificaciones de entrada y salida y **alcanza el 100% de la cobertura de decisión con el mínimo número de casos de prueba?** Suponga que la entrada es el primer número en cada par y la salida es el segundo.

- A. 0, 1; 1, 1; 2, 2.
- B. -1, -1; 1, 1; 5, 120.
- C. -1, -1; 0, 1; 1, 1; 3, 6.
- D. -1, -1; 0, 1; 4, 12.

107. Objetivo del aprendizaje: LO-4.4.4
Considere la siguiente función:

```
double interest(double avg_balance, double annual_rate)
/* Calcula el interés mensual de una cuenta corriente
/* que genera interés, basándose en el interés del
/* saldo promedio mensual. Si el saldo promedio
/* mensual (avg_balance) es menor que cero entonces se
/* calcula un interés negativo, pero otros módulos
/* controlan los cargos por sobregiro.
/* La tasa anual se expresa en porcentajes. */
{
    double calc_int=0;
    if (avg_balance > 0.0) {
        double monthly_rate = annual_rate/12.0;
        calc_int = avg_balance*(monthly_rate/100.0);
    }
    return(calc_int);
}
```

Suponga que tiene un arnés de pruebas que le permitirá presentar los valores de pruebas para la función del interés (las entradas) y para comprobar los valores que devuelve (las salidas), redondeado al céntimo más cercano. ¿Cuál de los siguientes casos de prueba alcanzan el **100% de la cobertura de sentencias con el minino número de pruebas?** Suponga que las entradas son los dos primeros números en cada triple y que la salida es el tercer número.

- A. 100.0, 5.0, 0.42.
- B. 100.0, 5.0, 0.42; -50.0, 1.25, 0.0.
- C. 100.0, 5.0, 0.42; 0.0, 2.5, 0.0; -50.0, 1.25, 0.0.
- D. 100.0, 5.0, 0.42; 0.01, 25.0, 0.0; 0.0, 2.5, 0.0; -0.01, 10.0, 0.0; -50.0, 1.25, 0.0.

108. Objetivo del aprendizaje: término
¿Qué es la cobertura de decisión?
- A. Una tabla que muestra las combinaciones de entradas con sus salidas asociadas.
 - B. Un punto del programa en el cual el flujo de control tiene dos o más rutas alternativas.
 - C. El porcentaje de sentencias ejecutables que han sido ejercidas por un juego de pruebas.
 - D. El porcentaje de los resultados de las decisiones que han sido ejercidas por un juego de pruebas.

Sección 4.5 Técnicas basadas en la experiencia (K2)

Términos

Pruebas exploratorias y ataque de fallas.

#	Pregunta	K
109.	Objetivo del aprendizaje: LO-4.5.1 ¿Cuál de las siguientes es una buena razón para utilizar las técnicas basadas en la experiencia para las pruebas?	1
110.	Objetivo del aprendizaje: LO-4.5.2	2

¿En cuál de las siguientes situaciones debe usted confiar más fuertemente en las técnicas basadas en la experiencia que en las técnicas basadas en la especificación?

- A. Le han dado a usted las especificaciones de los requisitos y del diseño, y tiempo adecuado para prepararse.
- B. Usted no tiene ninguna documentación escrita acerca de cómo debería funcionar el sistema.
- C. Usted está liderando un equipo de personas nuevas en el dominio del negocio y la tecnología del proyecto.
- D. Le han dicho a usted que prevenga defectos a través de una planificación, un análisis y diseño por adelantado.

111.	<p>Objetivo del aprendizaje: término ¿Qué son las pruebas exploratorias?</p> <ul style="list-style-type: none"> A. Un método de prueba en el cual el juego de pruebas se compone de todas las combinaciones de los valores de entrada y las precondiciones. B. Una técnica informal del diseño de pruebas donde el probador controla activamente el diseño de las pruebas a medida que esas pruebas son realizadas, y utiliza la información obtenida durante las pruebas para diseñar nuevas y mejores pruebas. C. Las pruebas llevadas a cabo informalmente; no se necesita ninguna preparación formal acerca de las pruebas, no se utiliza ninguna técnica reconocida de diseño de pruebas, no hay resultados esperados y la ejecución de las pruebas es una actividad aleatoria. D. Las pruebas simuladas o reales operacionales por usuarios/clientes o un equipo de pruebas independiente en el sitio de los desarrolladores, pero fuera de la organización del desarrollo. 	1
------	---	---

Sección 4.6 Selección de las técnicas de pruebas (K2)

Términos

Sin términos específicos.

#	Pregunta	K
112.	<p>Objetivo del aprendizaje: LO-4.6.1</p> <p>Usted está probando un dispositivo médico controlado por un software, de seguridad crítica, que será implantado en los cuerpos de los pacientes. La falla del dispositivo, incluyendo la falla del software relacionado, significa que el paciente puede morir. Cualquier anomalía con el dispositivo requeriría cirugía invasiva y cirugía peligrosa para retirar y reemplazar el dispositivo y/o su software. ¿Cuáles son las técnicas apropiadas que se deberían utilizar?</p> <ul style="list-style-type: none"> A. Las basadas en la estructura y las basadas en la especificación. B. Las basadas en la estructura, las basadas en la especificación y las basadas en la experiencia. C. Todas las técnicas dinámicas y estáticas disponibles. D. Las basadas en la experiencia. 	2

Capítulo 4 Pregunta de todas las secciones

#	Pregunta	K
113.	<p>Cubre: Secciones 4.2, 4.3 y 4.4.</p> <p>¿Cuál de los siguientes es un método basado en la estructura para el diseño de los casos de prueba?</p> <ul style="list-style-type: none"> A. Diagramas de transición de estados. B. Análisis de valores límite. C. Particionamiento de equivalencias. D. Cobertura de sentencias. 	1

Preguntas del Examen de Simulación 1

#	Pregunta	K
114.	<p>Objetivo del aprendizaje: LO-4.1.3</p> <p>Usted está trabajando en un proyecto para construir una aplicación bancaria en</p>	2

	<p>línea. Considere el siguiente extracto de la especificación de los requisitos:</p> <p>El sistema debe permitir al cliente tres intentos para ingresar una identificación de usuario y palabra clave válidos en la pantalla de bienvenida. Si han sido ingresadas tres veces las combinaciones de la identificación de usuario/palabra clave inválidas, el sistema debe bloquear temporalmente la cuenta del usuario.</p> <p>Usted ha escrito una especificación del diseño de pruebas que incluye, entre otros, las siguientes dos condiciones de prueba:</p> <ul style="list-style-type: none"> • Probar el ingreso exitoso en el sistema por medio de la identificación del usuario/clave del usuario con: cero intentos fallidos antes de tener éxito; un intento fallido antes de tener éxito; y dos intentos fallidos antes de tener éxito. • Probar el ingreso fallido al sistema por medio de la identificación de usuario/palabra clave <p>¿Cuál de los siguientes es un conjunto de casos de prueba que tiene una clara trazabilidad a, y una completa cobertura de, exactamente una de las condiciones de prueba enumeradas? Suponga que las entradas son los dos primeros ítems en cada triplete, y el resultado esperado es el tercero.</p> <ol style="list-style-type: none"> A. prueba0, válido0, éxito. B. prueba0, válido0, éxito; prueba1, inválido1, falla; prueba1, válido1, éxito; prueba2, inválido2, falla; prueba2, inválido2, falla; prueba2, válido2, éxito. C. prueba1, inválido1, falla; prueba1, válido1, éxito. D. prueba1, inválido1, falla; prueba1, inválido1, falla. 	
115.	<p>Objetivo del aprendizaje: LO-4.1.1</p> <p>¿Cuál de las siguientes es una diferencia entre los contenidos de una especificación de un caso de prueba y una especificación de un procedimiento de prueba?</p> <ol style="list-style-type: none"> A. La especificación del procedimiento de prueba especifica la secuencia de acciones para la ejecución de una prueba. B. La especificación del caso de prueba especifica la secuencia de las acciones para la ejecución de una prueba. C. Los dos términos son sinónimos y tienen exactamente los mismos contenidos. D. La especificación del procedimiento de prueba es utilizada solamente para la ejecución de las pruebas automatizadas. 	1
116.	<p>Objetivo del aprendizaje: LO-4.2.2</p> <p>Usted está trabajando en las pruebas de una aplicación e-learning. Un analista de negocios le entrega a usted un documento que describe los escenarios más comunes de los usuarios. Usted utilizaría este documento para crear ¿cuál tipo de pruebas?</p> <ol style="list-style-type: none"> A. Prueba basada en la estructura. B. Prueba basada en la experiencia. C. Prueba de componente. D. Prueba basada en la especificación. 	2
117.	<p>Objetivo del Aprendizaje: LO-4.3.1</p> <p>Usted está probando un sistema de comercio electrónico que vende provisiones de cocina tales como especias, harina y otros artículos al por mayor. Las unidades en las que los artículos son vendidos, son ya sea gramos (para especias u otros artículos costosos) o kilogramos (para harina y otros artículos económicos). Sin importar las unidades, la cantidad válida del pedido más pequeño es de 0.5 unidades (p.ej., medio gramo de vainas de cardamomo) y la cantidad válida más grande es de 25.0 unidades (p.ej., 25 kilogramos de azúcar). La precisión del campo de las unidades es de 0.1 unidades.</p> <p>¿Cuál de los siguientes es un conjunto de valores de entrada que cubren las particiones de equivalencia para este campo?</p> <ol style="list-style-type: none"> A. 0.4, 0.5, 25.0, 25.1 B. 10.0, 28.0 C. 0.2, 0.9, 29.5 D. 12.3 	3
118.	<p>Objetivo del Aprendizaje: LO-4.3.1</p> <p>Usted está probando un surtidor de gasolina que solamente acepta tarjetas de crédito y es no asistido. Una vez que la tarjeta de crédito es validada, el cliente ha elegido el grado de la carga deseada, y el surtidor está listo para bombeo, el cliente puede cancelar la transacción y no deber nada; sin embargo una vez que el bombeo comienza, la gasolina será vendida en centésimas (0,01) de galón. El surtidor continúa bombeando gasolina hasta que el usuario para o se llega a un</p>	3

	<p>máximo de 50,00 galones.</p> <p>¿Cuál de las siguientes es un conjunto mínimo de transacciones de la compra (en galones de gasolina repartida) que cubra los valores límite para esta variable?</p> <ol style="list-style-type: none"> 0.00, 20.00 0.00, 0.01, 50.00, 50.01 0.00, 0.01, 50.00, 70.00 -0.01, 0.00, 0.01, 25.00, 49.99, 50.00, 50.01, 75.00 																																	
119.	<p>Objetivo del Aprendizaje: LO-4.3.1</p> <p>Usted está probando un subsistema bancario que proporciona protección contra el sobregiro para los clientes quienes tienen esta característica para sus cuentas corrientes. La protección contra el sobregiro permite a los clientes sobregirar temporalmente su saldo de la cuenta, con cierto límite de crédito predefinido, sin tener cheques devueltos sin pagar. Una parte de la tabla de decisión que describe esa característica es mostrada abajo.</p> <table border="1"> <thead> <tr> <th colspan="2"><u>Condición</u></th> <th colspan="2"><u>Acción</u></th> </tr> </thead> <tbody> <tr> <td>Saldo excedido</td> <td>No</td> <td>Si</td> <td>Si</td> </tr> <tr> <td>Límite de crédito OK.</td> <td>No</td> <td>Si</td> <td>No</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <th colspan="2"><u>Acción</u></th> <th colspan="2"><u>Condición</u></th> </tr> <tr> <td>Aceptar cheque</td> <td>Si</td> <td>Si</td> <td>No</td> </tr> <tr> <td>Retornar cheque</td> <td>No</td> <td>No</td> <td>Si</td> </tr> <tr> <td>Enviar aviso</td> <td>No</td> <td>Si</td> <td>Si</td> </tr> </tbody> </table> <p>¿Cuántas pruebas diseñaría usted para cubrir la tabla de decisiones?</p> <ol style="list-style-type: none"> 3 2 5 15 	<u>Condición</u>		<u>Acción</u>		Saldo excedido	No	Si	Si	Límite de crédito OK.	No	Si	No					<u>Acción</u>		<u>Condición</u>		Aceptar cheque	Si	Si	No	Retornar cheque	No	No	Si	Enviar aviso	No	Si	Si	3
<u>Condición</u>		<u>Acción</u>																																
Saldo excedido	No	Si	Si																															
Límite de crédito OK.	No	Si	No																															
<u>Acción</u>		<u>Condición</u>																																
Aceptar cheque	Si	Si	No																															
Retornar cheque	No	No	Si																															
Enviar aviso	No	Si	Si																															
120.	<p>Objetivo del Aprendizaje: LO-4.3.1</p> <p>Usted está probando un automóvil con un interruptor de encendido/apagado controlado por software para el motor. El motor tiene dos estados, <i>funcionando</i> y <i>no funcionando</i>. Hay dos eventos que pueden ocurrir, una señal de <i>encendido</i> y una señal de <i>apagado</i>.</p> <p>Si el motor no está funcionando, presionando el interruptor de encendido/apagado envía al software una señal de encendido que indica al software que intente (hasta cinco segundos) poner en marcha el motor. Si el motor falla en ponerse en marcha, el interruptor de encendido/apagado puede ser presionado de nuevo para volver a intentar la operación tantas veces como quiera el conductor. En otras palabras hay dos condiciones, <i>éxito</i> y <i>fracaso</i>, que influencian el estado del motor que da como resultado y la acción tomada por el software basado en la señal de <i>encendido</i>.</p> <p>Si el motor está funcionando, el interruptor de encendido/apagado le indica al software que detenga el motor inmediatamente de manera segura. Si el motor no puede ser detenido de manera segura, el software le dará al conductor una advertencia verbal: "El motor no se puede apagar de manera segura". En otras palabras, hay dos condiciones <i>seguras</i> e <i>inseguras</i>, que influencian el estado del resultado del motor y la acción tomada por el software basado en la señal de <i>apagado</i>.</p> <p>Suponga que quiere describir este comportamiento en una <i>tabla de transición de estados</i> para diseñar un conjunto de pruebas para ambas situaciones <i>válidas</i> e <i>inválidas</i>. Suponga que cada fila en la tabla da el estado inicial, la combinación evento/condición, el estado resultante y la acción tomada. ¿Cuántas filas tendrá esta tabla?</p> <ol style="list-style-type: none"> 2 4 6 8 	3																																
121.	<p>Objetivo del aprendizaje: LO-4.3.3</p> <p>Considere los siguientes niveles de pruebas abordados en el programa de estudios del ISQTB 2007 Nivel Básico</p> <ol style="list-style-type: none"> Prueba de componentes. Prueba de integración. Prueba de sistema. Prueba de aceptación. <p>¿Cuál de estos niveles son explícitamente abordados en el programa de estudios como un beneficio del diseño de pruebas basado en los casos de uso?</p> <ol style="list-style-type: none"> Todos los cuatro niveles. 	1																																

	B. II, III, y IV solamente. C. I, II, y III solamente. D. III solamente.	
122.	<p>Objetivo del aprendizaje: LO-4.4.3</p> <p>Consideré la siguiente función:</p> <pre>double interest(double avg_balance, double annual_rate) /* Calcula el interés mensual para una * cuenta corriente que genera interés, basando el * * interés en el saldo promedio. * * Si el saldo medio es menor a cero, es calculado * * un interés negativo, pero otros módulos * * tratan los cargos de sobreiro. * * La tasa anual es expresada como un porcentaje. */ { double calc_int=0.0; if (avg_balance > 0.0) { double monthly_rate = annual_rate/12.0; calc_int = avg_balance*(monthly_rate/100.0); } return(calc_int); }</pre> <p>Suponga que tiene un arnés de pruebas que le permitirá presentar los valores de prueba a la función de interés (las entradas) y para comprobar los valores que devuelve (las salidas), redondeados al céntimo más cercano. ¿Cuál de los siguientes casos de prueba alcanzan el 100% de la cobertura de decisión con el número mínimo de pruebas? Suponga que las entradas son los dos primeros números en cada triple y la salida es el tercero.</p> <p>A. 100.0, 5.0, 0.42. B. 100.0, 5.0, 0.42; -50.0, 1.25, 0.0. C. 100.0, 5.0, 0.42; 0.0, 2.5, 0.0; -50.0, 1.25, 0.0. D. 100.0, 5.0, 0.42; 0.01, 25.0, 0.0; 0.0, 2.5, 0.0; -0.01, 10.0, 0.0; -50.0, 1.25, 0.0.</p>	3
123.	<p>Objetivo del aprendizaje: LO-4.5.1</p> <p>¿Cuál de las siguientes no es una razón para probar basándose en la intuición, la experiencia y el conocimiento?</p> <p>A. Para aumentar las técnicas sistemáticas. B. Para identificar condiciones especiales de las pruebas. C. Para ir más allá de las pruebas creadas con técnicas formales. D. Para permitir a los probadores no expertos en participar en la ejecución de las pruebas.</p>	1
124.	<p>Objetivo del aprendizaje: término</p> <p>¿Qué es la predicción de errores?</p> <p>A. Una técnica de diseño de pruebas donde la experiencia del probador es utilizada para anticipar qué defectos podrían presentarse en el componente o sistema bajo pruebas como un resultado de los errores cometidos, y para diseñar las pruebas específicamente para exponerlos. B. Una técnica informal de diseño de pruebas donde el probador controla activamente el diseño de las pruebas a medida que esas pruebas son realizadas y utiliza la información obtenida mientras se está probando para diseñar pruebas nuevas y mejores. C. El proceso de añadir intencionalmente defectos conocidos a aquellos que ya se encuentran en el componente o sistema con el propósito de monitorear la tasa de detección y eliminación, y la estimación del número de defectos restantes. D. La capacidad de un sistema o componente de continuar una operación normal a pesar de la presencia de las entradas erróneas.</p>	1
125.	<p>Objetivo del aprendizaje: LO-4.6.1</p> <p>Usted es incorporado como el único probador en el final de un proyecto. Fue seleccionado por su comprensión del sistema y su comportamiento previsto. Los usuarios, los clientes y otros interesados del negocio en el proyecto consideran el software de bajo riesgo. Le dan una semana para ejecutar las pruebas para ver si hay algunos importantes defectos presentes. ¿Cuál de las siguientes es probablemente su técnica principal del diseño de pruebas?</p> <p>A. Basada en la estructura. B. Basada en la especificación. C. Basada en la experiencia. D. Pruebas estáticas.</p>	2

Preguntas del Examen de Simulación 2

#	Pregunta	K
126.	<p>Objetivo del aprendizaje: LO-4.1.1</p> <p>¿Qué contiene una especificación del diseño de pruebas?</p> <ul style="list-style-type: none"> A. Las entradas de las pruebas. B. Los resultados esperados. C. Los pasos de procedimiento. D. Las condiciones de prueba. 	1
127.	<p>Objetivo del aprendizaje: LO-4.1.3 3</p> <p>Usted está probando un sistema de comercio electrónico para comprar libros en línea. Un extracto de la especificación de los requisitos indica lo siguiente:</p> <p>El cliente podrá pedir desde 1 a 99 copias de cualquier libro que esté en el almacén y si es que hay suficientes copias en el almacén para satisfacer la cantidad pedida.</p> <p>Otras partes de la especificación de los requisitos se encargan del control de los montos inválidos de los pedidos, incluyendo los pedidos que exceden el almacén disponible.</p> <p>¿Cuál de los siguientes es un caso de prueba bien escrito para el extracto aquí proporcionado de la especificación de los requisitos?</p> <ul style="list-style-type: none"> A. Pedir 0 libros. B. Pedir 100 libros; espere un mensaje de error. C. Pedir 1 libro el cual está en el almacén; espere la aceptación del pedido. D. Pedir 1 libro el cual está en el almacén. 	3
128.	<p>Objetivo del aprendizaje: LO-4.1.2</p> <p>¿Qué es una condición de prueba? Un ítem o evento de un componente o sistema que puede ser verificado por uno o más casos de prueba.</p> <ul style="list-style-type: none"> B. Una condición o capacidad necesitada por el usuario para resolver un problema o lograr un objetivo, que debe ser cumplido o poseído por un sistema o un componente de sistema para satisfacer un contrato, un estándar, una especificación u otro documento formalmente impuesto. C. Un método para las pruebas en el cual los casos de prueba son diseñados basados en los objetivos de las pruebas y las condiciones de las pruebas derivadas de los requisitos. D. Un conjunto de los valores de entrada, las precondiciones de ejecución, los resultados esperados y las poscondiciones de ejecución. 	1
129.	<p>Objetivo del aprendizaje: LO-4.2.2</p> <p>Un tipo de técnica de diseño de pruebas le permite derivar casos de prueba sistemáticamente de los modelos del comportamiento de los sistemas, los cuales son determinados antes de la ejecución de las pruebas. ¿Cuál tipo de técnica de diseño de pruebas es esta?</p> <ul style="list-style-type: none"> A. Basada en la estructura. B. Basada en la especificación. C. Basada en la experiencia. D. Predicción de errores. 	2
130.	<p>Objetivo del aprendizaje: LO-4.3.1</p> <p>Usted está probando un surtidor de gasolina con que solamente acepta tarjetas de crédito y no es asistido. Una vez que la tarjeta de crédito es validada, el cliente ha elegido el grado de la carga deseada y el surtidor está listo para bombear, el cliente puede cancelar la transacción y no deber nada; sin embargo una vez que el bombeo comienza, la gasolina será vendida en centésimas (0,01) de galón. El surtidor continúa bombeando gasolina hasta que el usuario para o se llega a un máximo de 50,00 galones.</p> <p>¿Cuál de las siguientes es un conjunto mínimo de las transacciones de compra (en galones de gasolina repartida) que cubra las particiones de equivalencia para esta variable?</p> <ul style="list-style-type: none"> A. 0.00, 20.00, 60.00 B. 0.00, 0.01, 50.00 C. 0.00, 0.01, 50.00, 70.00 D. -0.01, 0.00, 0.01, 25.00, 49.99, 50.00, 50.01, 75.00 	3
131.	<p>Objetivo del aprendizaje: LO-4.3.1</p> <p>Usted está probando un sistema de comercio electrónico que vende provisiones</p>	3

de cocina así como especias, harina y otros artículos al por mayor. Las unidades en las que los artículos son vendidos, son ya sea gramos (para especias u otros artículos costosos) o kilogramos (para harina y otros artículos económicos). Sin importar las unidades, la cantidad válida del pedido más pequeña es de 0.5 unidades (p.ej., medio gramo de vainas de cardamomo) y la cantidad válida más grande es de 25.0 unidades (p.ej., 25 kilogramos de azúcar). La precisión del campo de unidades es de 0.1 unidades.

¿Cuál de los siguientes es un conjunto de los valores de entrada que cubren los **valores límite** para este campo?

- A. 10.0, 28.0
- B. 0.4, 0.5, 25.0, 25.1
- C. 0.2, 0.9, 29.5
- D. 12.3

132. Objetivo del aprendizaje: LO-4.3.1

Considere la siguiente tabla de decisión para la parte de un sistema en línea de reservas de una aerolínea que permite a los viajeros frecuentes canjear puntos por viajes de recompensa:

<u>Condición</u>			
Cuenta/PIN OK.	No	Si	Si
Puntos suficientes	-	No	Si
<u>Acción</u>			
Mostrar historial de vuelos	No	Si	Si
Permitir viaje de recompensa	No	No	Si

Suponga que hay dos particiones de equivalencia para la condición donde *Cuenta/PIN OK* no es verdadero, una donde la cuenta es inválida y otra donde la cuenta es válida pero el PIN es inválido. Suponga que solo hay una partición de equivalencia correspondiente a la condición donde *Cuenta/PIN OK* es verdadero y donde ambos la cuenta y el PIN son válidos.

Si quiere diseñar pruebas para cubrir las particiones de equivalencia para la condición *Cuenta/PIN OK* y también para esta parte de la tabla de decisión, ¿Cuántas pruebas necesita?

- A. 2
- B. 3
- C. 4
- D. 9

133. Objetivo del aprendizaje: LO-4.3.1

Considere el siguiente diagrama de transición de estados para un surtidor de gasolina que acepta tarjetas de crédito es no asistido:



Suponga que quiere desarrollar el mínimo número de pruebas para cubrir cada transición en el diagrama de transición de estados. Suponga además que cada prueba debe comenzar y terminar en el estado inicial, *Esperando por el cliente*. ¿Cuántas pruebas necesita?

- A. 4
- B. 7
- C. 1
- D. Infinitas.

134. Objetivo del aprendizaje: LO-4.4.3

Considere la siguiente función:

3

3

3

```

int factorial(int n)
/* Calcula un factorial utilizando recursión. *
 * Un factorial de un número es el producto *
 * del número mismo por el factorial *
 * del número menos 1; es decir,
 *   *
 *   *
 * n! = n*((n-1)!)
 * Los factoriales de 0 y 1 son ambos 1.   */
{
    int f=1;
    if (n < 0) {
        fprintf(stderr, "factorial: Argumento negativo.\n");
    } else if ((n == 0) || (n == 1)) {
        f=1;
    } else {
        f=n*factorial(--n);
    }
    return(f);
}

```

Suponga que tiene un arnés de pruebas que le permitirá pasar los valores de pruebas a la función factorial (las entradas) y comprobar los valores que devuelve (las salidas). ¿Cuál de los siguientes conjuntos de casos de prueba proporciona las especificaciones correctas de las entradas y salidas, y logra el **100% de la cobertura de sentencias con el mínimo número de casos de prueba?** Suponga que la entrada es el primer número en cada par y que la salida es el segundo.

- A. 0, 1; 1, 1; 2, 2.
- B. -1, -1; 1, 1; 5, 120.
- C. -1, -1; 0, 1; 1, 1; 3, 6.
- D. -1, -1; 0, 1; 4, 12.

135.	<p>Objetivo del aprendizaje: LO-4.5.2</p> <p>¿Cuál es la verdad acerca de las técnicas basadas en la experiencia?</p> <ul style="list-style-type: none"> A. Las pruebas son derivadas de la habilidad, intuición y experiencia del probador en aplicaciones y tecnologías similares. B. Las técnicas no son útiles para identificar algunas pruebas adicionales más allá de las capturadas fácilmente por las técnicas formales. C. Todas las pruebas que se derivan de las técnicas basadas en la experiencia son redundantes de las técnicas basadas en la especificación. D. Todas las pruebas que se derivan de las técnicas basadas en la experiencia son redundantes de las técnicas basadas en la estructura. 	1
136.	<p>Objetivo del aprendizaje: término</p> <p>¿Qué es la cobertura de decisión?</p> <ul style="list-style-type: none"> A. El porcentaje de los resultados de las condiciones que han sido ejercidas por un juego de pruebas. B. La cobertura de decisión es un sinónimo de la cobertura de sentencias. C. El porcentaje de las sentencias ejecutables que han sido ejercidas por un juego de pruebas. D. El porcentaje de los resultados de las decisiones que han sido ejercidas por un juego de pruebas. 	1
137.	<p>Objetivo del aprendizaje: LO-4.6.1</p> <p>Usted no tiene acceso al código o alguna otra información de caja blanca acerca de la implementación de un sistema. Usted tiene acceso a un conjunto extenso y bien escrito de los requisitos del usuario. Se le pide prevenir tantos defectos como sea posible antes de la entrega del sistema, así como también la mejora de la confianza en el sistema y la detección de los defectos una vez que el sistema sea entregado a usted. ¿Cuál de las siguientes técnicas de pruebas quisiera usted utilizar?</p> <ul style="list-style-type: none"> A. Todas las técnicas posibles basadas en la especificación. B. Todas las técnicas posibles basadas en la estructura. C. Todas las técnicas posibles basadas en la experiencia. D. Todas las posibles técnicas dinámicas y estáticas. 	2

23 Este ejercicio y su solución han sido adaptados del capítulo 7, del libro de Rex Black, *Pragmatic Software Testing*.

24 Este ejercicio y su solución han sido adaptados del capítulo 11, del libro de Rex Black, *Pragmatic Software Testing*.

25 Este ejercicio y su solución han sido adaptados del capítulo 13, del libro de Rex Black, *Pragmatic Software Testing*.

26 Este ejercicio y su solución han sido adaptados del capítulo 15, del libro de Rex Black, *Pragmatic Software Testing*.

27 Aunque no está definido en la Real Academia Española, es un verbo utilizado en la informática

que quiere decir dividir o partir algo en particiones. P.ej. un disco duro o en clases de equivalencia.

28 Este ejercicio y su solución han sido adaptados del capítulo 22, del libro de Rex Black, *Pragmatic Software Testing*.

Capítulo 5

Gestión de Pruebas

"No se puede controlar lo que no se puede medir"
Tom DeMarco, desarrollo del análisis estructurado

El capítulo 5, Gestión de Pruebas, contiene las siguientes 6 secciones:

1. Organización de pruebas.
2. Planificación y estimación de pruebas.
3. Monitoreo y control del progreso de pruebas.
4. Gestión de configuración.
5. Riesgo y pruebas.
6. Gestión de incidencias.

Cada una de las secciones estará desglosada en dos o más partes.

5.1 Organización de Pruebas

Objetivos del Aprendizaje

- LO-5.1.1 Reconocer la importancia de las pruebas independientes. (K1)
LO-5.1.2 Explicar los beneficios y las desventajas de las pruebas independientes dentro de una organización. (K2)
LO-5.1.3 Reconocer los diferentes miembros del equipo para que sean considerados para la creación de un equipo de pruebas. (K1)
LO-5.1.4 Recordar las tareas del típico líder de pruebas y el probador. (K1)

Glosario del ISTQB

Probador: Un profesional hábil quién está involucrado en las pruebas de un componente o sistema.

Líder de pruebas: Véase jefe de pruebas.

Gestión de pruebas: La planificación, la estimación, el monitoreo y el control de las actividades de las pruebas, típicamente realizadas por un jefe de pruebas. Note que este término (gestión de pruebas) no está siendo enunciado específicamente en esta sección pero es incluido aquí porque es esencial para comprender los términos control de pruebas y monitoreo de pruebas.

Esta sección, Organización de pruebas, cubrirá los siguientes conceptos clave:

- La importancia de las pruebas independientes.
- Los beneficios y las desventajas de las pruebas independientes.
- Los diferentes miembros del equipo que deben ser considerados para la organización de un equipo de pruebas.
- Las tareas de un típico líder de pruebas o probador.

Don Quijote: ¿Campeón solitario de Calidad? Usted puede hacerse un gran daño político, si no entiende su rol en la compañía.



Figura 5.1: ¿Cuál es el Trabajo de un Equipo de Pruebas?

Una pregunta importante—de hecho, en muchos casos, crítica—que es necesario de aclarar es, “¿Cuál es el trabajo o la misión del equipo de pruebas con respecto a la organización, el proyecto y

el producto?" Demasiado a menudo, esta pregunta queda sin ser contestada en las organizaciones, conduciendo a una enorme cantidad de ineficiencia y conflicto.

Entonces, ¿Cuáles servicios puede ofrecer un equipo de pruebas a una organización, a un proyecto y a un producto?

Bien, podemos probar el software o los sistemas que se nos ha entregado. Recuerde que la definición en el glosario del término "pruebas" es: "El proceso que consiste en todas las actividades del ciclo de vida, ambas estáticas y dinámicas, preocupadas por la planificación, la preparación y la evaluación de los productos de software y los productos del trabajo relacionados, para determinar que ellos satisfagan los requisitos especificados, para demostrar que ellos están aptos para el propósito y para detectar defectos". Esto es un gran trabajo. De hecho, en la mayoría de las organizaciones y los proyectos, ese trabajo necesita ser distribuido en parte al desarrollo por lo menos, las pruebas de unidad y las revisiones del código. A menudo, un equipo de pruebas independiente se enfoca en un sólo nivel de pruebas o dos niveles relacionados lógicamente; p.ej. Las pruebas de sistema y las pruebas de integración de sistemas. Ésta es una misión alcanzable, con la condición de que cada uno comprenda que no podemos ser 100% efectivos en ninguna de las siguientes 3 actividades—los requisitos de las pruebas, la comprobación de la aptitud para el propósito o los defectos detectados. La imposibilidad de las pruebas exhaustivas limita nuestra efectividad a menos del 100% en cualquiera de estas áreas y las restricciones del proyecto relacionadas con el tiempo y dinero la limitarán aún más.

Cuando trabajamos con clientes hemos notado que ellos se refieren acerca de sus equipos de pruebas como grupos de "control de calidad". No hay una definición en el ISTQB para este término, pero hay una definición estándar del Japón que denomina el control de calidad como "El sistema de medios para producir económicamente productos o servicios que satisfagan los requisitos del cliente".

Tradicionalmente, el control de calidad en la industria manufacturera destinó los procesos para detectar la inconformidad con los requisitos especificados con precisión, así como el ancho de las cabezas de los tornillos o la reflectividad de una superficie pulida. Eso funciona bien para la parte del hardware de un proyecto de desarrollo de sistemas—p.ej., la detección del número de pixeles defectuosos en una pantalla LCD y el rechazo de las pantallas con más de tres pixeles defectuosos —pero es más difícil para el software. Por un lado es rara la vez que un software sea, preciso, no tenga ambigüedades y tenga las especificaciones completas de los requisitos.

Por otro lado, la satisfacción con el software puede depender frecuentemente de un número de intangibles que están más allá de nuestra capacidad de probar, incluyendo atributos de la adquisición, la entrega y los procesos del soporte. Usted es un jefe de pruebas valiente que permite la evaluación del rendimiento de su equipo por medio de las encuestas acerca de la satisfacción de los clientes.

Glosario del ISTQB

Jefe de Pruebas: La persona responsable para la gestión del proyecto de las actividades y los recursos de las pruebas y la evaluación de un objeto de prueba. El individuo quién dirige, controla, administra, planifica y regula la evaluación de un objeto de prueba.

En el capítulo 4, aprendió cómo utilizar el análisis de los riesgos de calidad para basar los casos de prueba en los riesgos y para optimizar la secuencia y el esfuerzo de las pruebas basadas en los niveles de riesgo. Esto quiere decir que las pruebas pueden ofrecer un servicio de gestión de los riesgos de calidad para una organización. De hecho, podemos medir el nivel residual del riesgo e informar nuestros resultados basados en eso. (Más acerca de eso en una sección más adelante de este capítulo).

Podemos también ofrecer nuestros servicios como asesores de calidad. Podemos decir que la calidad implica ambos la conformidad con los requisitos especificados y la capacidad del sistema de satisfacer al usuario, el cliente, y las necesidades y expectativas de los interesados. Por supuesto, no podemos hacer esto perfectamente, porque tiene que haber por seguro alguna prueba que podríamos haber ejecutado relacionada con algún usuario u otro que no lo vamos a conseguir.

Ahora, los problemas reales comienzan cuando los grupos aceptan la carta ("charter") del "aseguramiento de la calidad". El aseguramiento de la calidad se define según el Glosario como "La parte de la gestión de la calidad enfocada en proveer la confianza en que los requisitos de calidad se cumplan", suena suficientemente tranquilo. Sin embargo, la mayoría de la gente utiliza la definición inglesa más común de "asegurar" cuando ellos piensan en el aseguramiento de la calidad.

De este modo esta definición dice que la misión del equipo del aseguramiento de calidad es "prevenir riesgos relacionados con la calidad" o "asegurar o cerciorarse de la calidad", "informar positivamente que la calidad está presente", "asegurar el logro de calidad", o, tal vez más comúnmente y más tóxicamente para la existencia continua del equipo de pruebas o "para garantizar la calidad". Un grupo de pruebas, especialmente uno comprometido en un nivel alto en el final del proyecto, no puede garantizar la calidad al igual que la compra de una balanza no puede garantizar la pérdida de peso. La calidad sale de una interacción compleja de las

características y los atributos del producto, implantada en todo el proceso—o no. Si no se controla el proceso entero de principio a fin, no puede asegurar la calidad. Sólo aquellos con el control de todo el proceso—p.ej., todo el equipo de gestión del proyecto—pueden asegurar la calidad. Francamente, a menudo la designación de un equipo como el “grupo del aseguramiento de calidad” es simplemente una forma para que el equipo de gestión del proyecto pueda apuntar a un culpable de los problemas de calidad cuando estos aparecen.

Asumiendo que su equipo tiene la misión de probar (o, tal vez del control de calidad), entonces usted tiene una necesidad de gestionar el proceso. La gestión del proceso significa, en este caso, un trabajo continuo para alinear los servicios de las pruebas que usted provee con las necesidades de la organización, el proyecto, y el producto, dentro de las restricciones que usted debe soportar, junto con el concepto más tradicional de gestión en cuanto a definir y llevar a cabo un plan para el conjunto de actividades.

No puede triunfar con una misión deficientemente definida, porque no importa qué bien hace su trabajo, alguien le dirá que no está haciendo su trabajo en absoluto. ¿Cómo puede argumentar ese punto si su trabajo no está definido? ¿Qué hará si su misión es insostenible? Una vez que obtenga una misión bien definida y sostenible, con el fin de triunfar, asegúrese de que usted y su equipo sean quienes lleven a cabo todas las tareas requeridas. Asegúrese de que tiene el personal adecuado. Y, especialmente si tiene un rol acerca del control del proceso como usted lo tendría con un equipo del aseguramiento de la calidad, asegúrese de tener un soporte político para ese rol.

Para ilustrar la importancia del soporte político para los cargos del Aseguramiento de la Calidad (“QA”), permítanos contarle la triste historia de un Jefe del Aseguramiento de la Calidad. Para proteger a la inocente—y ella era inocente—llamémosla Natasha.

Algunas veces, nuestro trabajo nos lleva a lugares hermosos. Nuestros asociados y nosotros hicimos un proyecto en Paris, Francia, justo en la primavera. Estuvimos allí para ayudar al Jefe de Desarrollo en la construcción de un framework de pruebas de unidad automatizadas para su equipo. Cuando llegamos, sólo para conocer quién era quién y quién más podría ser capaz de utilizar nuestro framework, hablamos con varias personas acerca de la organización y sus desafíos.

Preguntamos al Jefe de Desarrollo—llamémoslo Boris—qué él pensaba acerca de Natasha y su equipo.

“Natasha”, el dijo, “es una diletante narcisista”. El no quiso decir eso como un complemento.

“¿Cómo así?” preguntamos.

“Sus operaciones de pruebas son un completo desastre. Nadie sabe que está ocurriendo allí. Todavía ella encuentra tiempo para venir y sermonear a mis desarrolladores y líderes del desarrollo acerca de cómo ellos pueden realizar mejores pruebas de unidad y revisiones de código. Todos ellos se ponen tensos por sus lecciones pretenciosas—ella no sabe nada acerca de codificar —y tengo que pasar mucho tiempo calmando sus frustraciones”.

¿Por qué estaba Natasha haciendo esto? Bueno, resultó que cuando ella tomó el trabajo de Jefe del Aseguramiento de la Calidad (“QA”), no había una misión definida. Ella nunca había sido antes una Jefa del Aseguramiento de la Calidad (“QA”), así que ella le preguntó al Vicepresidente a cargo de su grupo (y también a cargo del grupo de Boris, a propósito) qué ella debería hacer. El dijo, “Bueno, haga Aseguramiento de la Calidad (“QA”).” Entonces ella fue y compró un libro acerca del Aseguramiento de la Calidad (“QA”). Éste libro afirmaba acerca del Aseguramiento de la Calidad (“QA”) como una tarea principalmente preventiva, enfatizando los conceptos de Gestión Total de la Calidad. Ella preguntó luego a su jefe, el Vicepresidente, si esa era la misión correcta. El dijo, “Seguro, lo que sea que el libro diga”.

Aquí viene el problema. Natasha olvidó de hacer que la misión sea claramente definida y comunicada a cada uno que es parte del organigrama del Vicepresidente. De este modo, ella no tuvo apoyo político para hacer eso. Ella agravó el problema empleando a alguien que no era técnico—es decir, ella misma—para interactuar con los desarrolladores acerca de su parte del trabajo del aseguramiento de la calidad. Al respecto, ella no tenía nada que ofrecerles sino más bien frases trilladas e intimidantes.

Natasha fue más tarde despedida. Pobre Natasha.

Hablamos acerca del concepto de la independencia de las pruebas en el capítulo uno. Revisemos los niveles de independencia aquí:

- En el nivel más bajo de independencia, las pruebas son realizadas por el autor del ítem de prueba.
- Las pruebas son realizadas por otra persona o gente dentro del mismo equipo.
- Las pruebas son realizadas por una persona o gente de un equipo diferente o por especialistas de pruebas.
- En el nivel más alto de independencia, las pruebas son realizadas por una persona o gente de una organización diferente o compañía.

¿Cuál es el valor de las pruebas independientes?

Bueno, comencemos a pensar acerca de lo que las pruebas pueden hacer con respecto a los defectos. Las pruebas pueden detectar algún porcentaje de los defectos y así ofrecer a la organización la oportunidad de mejorar la calidad.

En otras palabras, las pruebas son como un filtro, capturando un cierto porcentaje del material no deseado que fluye a través de éstas. En importantes circunstancias como la purificación del agua, los sistemas de agua municipales utilizan múltiples niveles de filtración.

Entonces, para las aplicaciones complejas y críticas, necesitamos múltiples niveles de pruebas. Además necesitamos alguno de esos filtros que sean bastante efectivos en detectar defectos, si es que queremos tener un número muy bajo de defectos, que fluyen a través del proceso y hacia los usuarios y clientes.

Las pruebas independientes tienden a ser más efectivas que las autopruebas en encontrar defectos.

Como regla general, lo que observamos en nuestros mejores clientes es que las pruebas son poco independientes en los niveles de pruebas más bajos. Para las pruebas de unidad es necesaria la amplia comprensión de la implementación de una unidad. El desarrollador—presumiblemente—ya ha adquirido la compresión en el proceso de implementación. Sin embargo, el autor tiene también una parcialización para creer que su solución es correcta—sino él no la hubiera implementado de esa manera. Los estudios de Capers Jones indican que los porcentajes de la detección de defectos para las pruebas de unidad oscilan entre el 10%(con las peores prácticas) y el 25%(con las prácticas típicas) y el 50%(con las mejoras prácticas). Las mejores prácticas incluirían la utilización de frameworks automatizados de las pruebas de unidad, acerca de las cuales conversaremos en el capítulo 6.

Nuestros mejores clientes realmente reservan las pruebas independientes para los altos niveles de las pruebas, así como la prueba de sistema, la prueba de integración de sistemas y la prueba de aceptación. Los equipos de pruebas independientes realizan la prueba de sistema, y, si es aplicable, la prueba de integración de los sistemas, mientras que los usuarios y los clientes realizan las pruebas de aceptación. Nuestra evaluación de los clientes demuestra que los porcentajes promedios de la detección de los defectos para los equipos independientes de pruebas oscilan alrededor del 85%(con las prácticas típicas), con el 95% y arriba (con las mejores prácticas).

¿Entonces qué pasa con el concepto de los equipos de pruebas independientes que promulgan las mejores prácticas en las pruebas tempranas, en los primeros niveles de pruebas? Bueno, al igual que la argumentación acerca de una posible misión del Aseguramiento de la Calidad para su equipo, le aconsejamos que avance aquí cuidadosamente. Si usted quisiera—o si le es concedida—la autoridad para exigir y definir los procesos y las reglas para aquellos procesos—ya sea para las pruebas de unidad o para alguna otra actividad temprana relacionada con la calidad—querrá una dirección clara y un apoyo político.

Entonces, arriba y más allá de un mejor nivel de detección de defectos, ¿Cuáles otros beneficios ofrecen las pruebas independientes?

Los probadores independientes no sólo tienden a ver más defectos, ellos también ven otros y diferentes defectos. Eso es, ellos tienden a tener un conjunto de ideas diferentes acerca de lo que la calidad significa, a menudo un sentido más amplio, que los desarrolladores individuales.

Además, los probadores independientes tienden a tener una actitud escéptica hacia el producto—la antítesis de la parcialidad del autor—que los hacen suponer que si hay alguna duda acerca del comportamiento observado, entonces es valioso de reportarlo como un defecto.

Esto hace que los probadores independientes calificados, sean participantes muy valiosos en las revisiones, desde los requisitos, pasando por el diseño hasta el código. Ellos tienden a preguntar—y así ayudan a verificar—las suposiciones embebidas en las especificaciones y la implementación.

Un equipo de pruebas independiente puede proveer una evaluación creíble de la calidad, lo que un desarrollador o un probador no podrían proveer dentro del equipo de desarrollo debido al riesgo de la edición o la autoedición del mensaje.

Finalmente, en un equipo de pruebas independiente, el probador tiene una trayectoria profesional. En otras palabras, los probadores tienen una razón para aprender a ser mejores probadores en vez de aprender a ser mejores desarrolladores.

Dicho eso, hay riesgos.

Es posible que el equipo de pruebas llegue a estar aislado del equipo de desarrollo. Especialmente esto puede pasar si el equipo de pruebas comienza a establecerse como un policía de la calidad o peor todavía, un policía del proceso.

Cuando las pruebas independientes vienen al final del proceso, usted puede escuchar algunas veces afirmaciones como, “Bueno, el proyecto está terminado solo que está siendo retrasado a causa de las pruebas”. Esto es generalmente inexacto, porque el número de defectos introducidos más temprano determina la duración de la ejecución de las pruebas tan frecuentemente como es el tiempo necesario para ejecutar las pruebas planificadas.

Finalmente, hay el riesgo de que las actividades tempranas de calidad como las pruebas de unidad se pongan peor cuando los programadores abandonen el sentido de la responsabilidad por la calidad. Esto resulta en una caída de la calidad del producto, no subida, porque las pruebas de alto nivel, las cuales son filtros sensitivos, a menudo son obstruidas y vencidas por la muy baja calidad y los muchos defectos del código.

El programa de estudios básico introduce el concepto de dos principales roles en las pruebas, el líder de pruebas y el probador. Revisemos las tareas que son asignadas a cada rol, comenzando por los líderes de pruebas.

Los líderes de pruebas llevan a cabo las siguientes tareas:

- Crear las estrategias y los planes de pruebas.
- Escribir o revisar las políticas de pruebas.
- Consultar acerca de las pruebas para las otras actividades del proyecto, así como las pruebas de unidad.
- Estimar el tiempo, el dinero y los recursos requeridos para las pruebas.
- Adquirir los recursos de las pruebas.
- Liderar la especificación, la preparación, la implementación y la ejecución de las pruebas.
- Monitorear y controlar la ejecución de las pruebas.
- Adaptar el plan de pruebas durante la ejecución de las pruebas en base a los resultados de las pruebas, así como el ajuste del nivel de los riesgos en base a los defectos realmente observados.
- Asegurar la gestión de configuración del testware, incluyendo el guardado seguro de los valiosos casos de prueba, los guiones, las herramientas y los datos.
- Asegurar la trazabilidad de las pruebas hacia atrás a las bases de las pruebas.
- Medir el progreso de las pruebas, evaluando e informando acerca de la calidad de las pruebas y el producto.
- Planificar cualquier trabajo de automatización de pruebas que debe ser realizado.
- Seleccionar herramientas.
- Organizar las capacitaciones para los probadores.
- Asegurar la implementación de un entorno de las pruebas.
- Programar el cronograma de la preparación y la ejecución de las pruebas, a menudo en base a la prioridad de los riesgos.
- Escribir informes del resumen de las pruebas.

Glosario del ISTQB

Estrategia de pruebas: Una descripción de alto nivel de los niveles de pruebas que deben ser realizados y de las pruebas en aquellos niveles para una organización o programa (uno o más proyectos). Note que este término no es específicamente utilizado en esta sección pero está aquí incluido porque es esencial para comprender el término método de prueba.

Ahora, los probadores llevan a cabo las siguientes tareas:

- Revisar y contribuir a los planes de pruebas.
- Analizar, revisar y evaluar los requisitos de los usuarios y otras especificaciones.
- Crear los juegos de pruebas, los casos de prueba, los datos de prueba y los procedimientos de prueba.
- Instalar el entorno de las pruebas.
- Implementar las pruebas de todos los niveles de pruebas (lo que a propósito significa que el desarrollador está jugando el rol de probador cuando está realizando las pruebas de unidad).
- Ejecutar y registrar las pruebas, evaluando los resultados y documentando algunos problemas encontrados.
- Monitorear las pruebas utilizando las herramientas apropiadas.
- Automatizar las pruebas.
- Medir el rendimiento de los componentes y los sistemas.
- Revisar cada uno los informes de las pruebas e incidencias.

En los proyectos de pruebas de Business Innovations/RBCS, típicamente refinamos el rol del probador:

En la mayoría de los niveles senior, tenemos ingenieros de pruebas. Estos son pares técnicos de programadores, gente que elige las pruebas como una especialidad. Ellos escriben casos de prueba y organizan juegos de prueba. Ellos crearán, personalizarán y utilizarán las herramientas avanzadas de pruebas, incluyendo la automatización de las pruebas. Ellos tienden a tener habilidades únicas de las pruebas.

En el nivel más bajo, tenemos los técnicos de pruebas. Nos agrada que ellos sean probadores hábiles y con experiencia, pero algunas veces ellos son nuevos en el campo. Idealmente, ellos aspiran a ser ingenieros de pruebas. Sus principales tareas son ejecutar las pruebas, informar los defectos y actualizar el estado de las pruebas. Ellos asisten a los ingenieros de pruebas.

En muchos equipos de pruebas, tenemos otros miembros del equipo de pruebas así como un administrador de sistemas o base de datos o ingenieros de versión o configuración. Cuando

tenemos que crear herramientas de pruebas, tenemos una posición llamada toolsmith29 de pruebas.

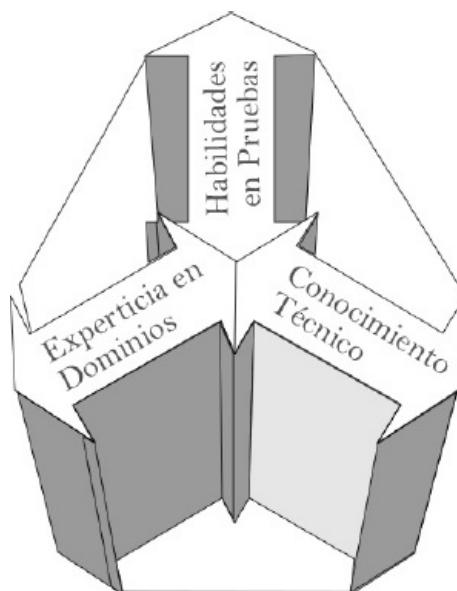
Los cargos correctos para su equipo dependen de las habilidades que necesite, lo cual es un asunto del cual hablaremos más en un momento. Necesita equilibrar las habilidades y los cargos a través de todo el equipo de pruebas. También, recuerde que las habilidades son complementarias y de que el todo puede ser más grande que la suma de las partes—o menos cuando las habilidades críticas hacen falta en **todos** los miembros del equipo de pruebas.

En muchos proyectos, los probadores aficionados son empleados como parte(o todos) de un equipo de pruebas. Un probador aficionado es definido como alguien que no prueba para vivir, pero es oficialmente empleado como un usuario del sistema, un jefe de proyecto, un jefe de calidad, un programador, un experto comercial o del dominio o un operador de infraestructura o TI.

Mientras que no tenemos problemas con los probadores aficionados que participan en algunas actividades de pruebas—especialmente en dominios de negocios o tecnologías complejos. Sin embargo, mientras un equipo se conformó exclusivamente de probadores aficionados, ellos tendrán a menudo habilidades fuertes en algunas áreas y habilidades muy débiles en otras áreas.

Por ejemplo, los programadores son a menudo fuertes técnicamente pero más débiles en el conocimiento del dominio del negocio, mientras los usuarios tienden a ser técnicamente débiles pero fuertes en el conocimiento del dominio de los negocios. En ambas formas, la mayoría de los probadores aficionados no tienen habilidades o experiencia substancial en las pruebas.

Las pruebas son un campo especial, con habilidades especiales. Los conceptos básicos son cubiertos en este libro, pero se necesitan aún tres libros avanzados aún más extensos que este libro antes de que usted haya sido expuesto a la mayoría de las prácticas comúnmente utilizadas en las pruebas. Un aficionado no sabrá ni los conceptos básicos, y eso conduce a muchos errores en las pruebas de principiante y que pueden ser fácilmente evitables.



La profundidad y el largo apropiado de cada flecha en la figura dependen del proyecto, el proceso y el producto

Figura 5.2: Equilibrio de las Habilidades

Los equipos buenos de pruebas tienen la mezcla correcta de las habilidades basadas en las tareas y actividades. ¿Entonces, cuáles son las habilidades necesarias de un probador o líder de pruebas? Estas se dividen en tres áreas principales:

- Dominio de la aplicación o del negocio: Éste se refiere a la comprensión del comportamiento previsto y el problema del negocio que tiene que ser resuelto.
- Pruebas: Esto es conocer los temas cubiertos en este libro y otros libros avanzados de pruebas.
- Tecnología: Esto es la conciencia de los asuntos y las limitaciones técnicas, la cual es útil para el análisis de los riesgos, el diseño de pruebas y el aislamiento de las incidencias.

Sería muy bien si pudiéramos entregarle el Currículum del Probador de Oro, un conjunto de habilidades y experiencias que constituirían las características del probador con habilidades perfectas. Sin embargo, las habilidades necesarias tienden a variar de un proyecto a otro, de un producto a otro, de una organización a otra. Piense acerca de esto: ¿Cuál es la mezcla correcta para cada uno de los siguientes proyectos?

Desarrollo un dispositivo de Internet.

Mantenimiento de un equipo médico que administra la terapia de radiación nuclear.

Su proyecto actual.

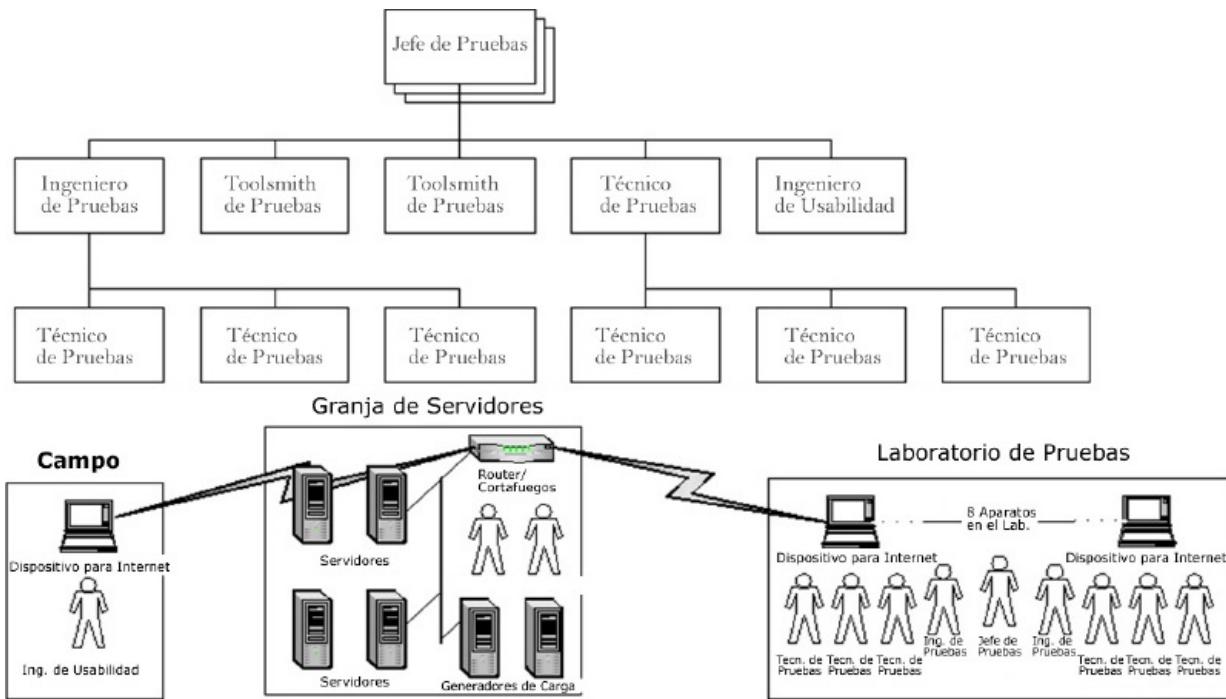


Figura 5.3: Caso de Estudio: Roles, Organización y Proyecto

Esta figura muestra la organización RBCS contratada para formar un equipo de pruebas independiente para un proyecto de un dispositivo de internet.

Un ingeniero de pruebas dirigió un equipo de tres técnicos en las pruebas del lado del cliente. En otras palabras, el equipo fue entregado efectivamente a los clientes.

Otro ingeniero de pruebas dirigió un equipo de tres técnicos en las pruebas del lado del servidor. En otras palabras, los sistemas finales de la parte de atrás en el centro de datos que permiten a los dispositivos enviar y recibir e-mail, navegar la Web, etc.

También tuvimos dos toolsmiths de pruebas que construyeron generadores de carga y otras herramientas de pruebas, principalmente para las pruebas del lado del servidor.

Finalmente, tuvimos un ingeniero de pruebas de usabilidad quién realizaba las pruebas de usabilidad de campo del dispositivo o de los prototipos de éste.

Todo el equipo nos informaba, porque Rex Black era el Jefe de Pruebas.

Como puede ver, el equipo de pruebas está bien alineado con la estructura del proyecto.

5.1.1 Ejercicios

Ejercicio 1

Imagine que usted es el jefe de pruebas para el proyecto Omninet, a cargo de las pruebas de sistema e integración.

Esquematice un organigrama que muestre como organizaría el equipo de pruebas de Omninet.

¿Dónde encajaría el equipo de pruebas en el equipo del proyecto de Omninet?

Argumente.

Solución del Ejercicio 1

La figura 5.4 muestra una posible estructura organizacional para el equipo de pruebas. Hay características notables en esta figura que llevan más explicación:

- Los probadores informan a un jefe de pruebas, no a un jefe de desarrollo, lo cual aumenta la independencia.
- El jefe de pruebas informa en última instancia a la persona responsable del ciclo de vida entero del producto, en vez de a un jefe cuyos incentivos se centran principalmente en las fechas de entrega y los presupuestos, lo que también fomenta la independencia de los equipos de pruebas así como también la credibilidad de los hallazgos.
- Hemos dividido el equipo de pruebas en tres grupos basados en la arquitectura del sistema. Porque el centro de datos, el centro de llamadas y el quiosco son tan diferentes, esta estructura ayudará a enfocar a los grupos. Hay el riesgo de omisiones y superposiciones entre

los grupos, especialmente en la falla potencial para realizar pruebas completas de punta a punta, entonces el jefe de pruebas necesitaría trabajar con tres grupos para mitigar ese riesgo.

- Tenemos dos técnicos de pruebas por cada ingeniero de pruebas. Estas personas asisten al ingeniero de pruebas de varias maneras, especialmente por medio de la ejecución de los casos de prueba manuales escritos. En la práctica, podríamos necesitar más o menos técnicos de pruebas por cada ingeniero de pruebas, lo cual sólo sabríamos una vez que hubiéramos puesto una estimación sólida de pruebas. Consideraríamos la rotación de los técnicos de pruebas entre los grupos para ayudar a reducir las omisiones y la superposición en las pruebas, así como también para asegurar la capacidad para manejar de manera flexible y temporal los aumentos repentinos de carga de trabajo en cualquier grupo.
- Los dos toolsmiths de pruebas son necesarios para crear herramientas de pruebas personalizadas y adaptar las herramientas de código abierto. No todos los tipos de pruebas que serán necesarias en Omninet pueden ser realizadas con herramientas comerciales.
- Tenemos un ingeniero de pruebas de usabilidad para enfocarnos en los aspectos de usabilidad del centro de llamadas y de los quioscos.

También hay otros métodos que funcionarían para este proyecto.

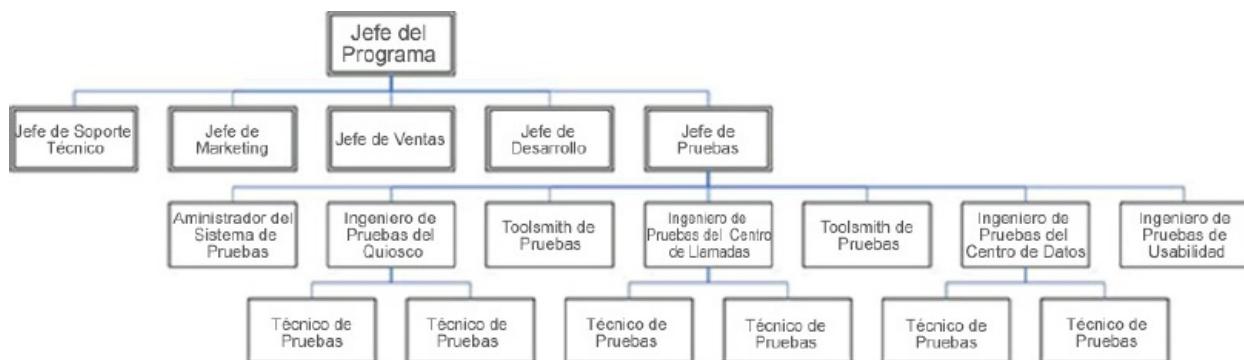


Figura 5.4: Organigrama del equipo de pruebas de Omninet.

Note que esto se parece mucho a la estructura organizacional del caso de estudio del Dispositivo de Internet, lo cual tiene sentido, porque Omninet es similar a ese proyecto. Cuando crea un equipo de pruebas específico para un proyecto, puede construir el equipo para reflejar las necesidades del proyecto. Para equipos que deben controlar una corriente continua de proyectos que tienen diferencias significativas, usaríamos una estructura más flexible y general.

5.2 Planificación y Estimación de Pruebas

Objetivos del Aprendizaje

- LO-5.2.1 Reconocer los diferentes niveles y objetivos de la planificación de las pruebas. (K1)
- LO-5.2.2 Resumir el propósito y el contenido del plan de pruebas, la especificación del diseño de pruebas y los documentos de los procedimientos de prueba de acuerdo con el "Estándar para la Documentación de las Pruebas de Software" (IEEE829). (K2)
- LO-5.2.3 Diferenciar entre los métodos conceptualmente diferentes, así como analíticos, los basados en modelos, los metódicos, los compatibles con un proceso o estándar, los dinámicos o heurísticos, los consultivos y los contrarios a la regresión. (K2)
- LO-5.2.4 Diferenciar entre el tema de la planificación de pruebas para un sistema y para el cronograma de la ejecución de las pruebas. (K2)
- LO-5.2.5 Escribir un cronograma de ejecución de pruebas para un conjunto dado de casos de prueba, considerando la priorización y las dependencias técnicas y lógicas. (K3)
- LO-5.2.6 Enumerar las actividades de la preparación y la ejecución de las pruebas que deben ser consideradas durante la planificación de las pruebas. (K1)
- LO-5.2.7 Recordar los factores típicos que influencian el esfuerzo relacionado con las pruebas. (K1)
- LO-5.2.8 Diferenciar entre dos métodos de estimación diferentes conceptualmente: el método basado en las métricas y el método basado en los expertos. (K2)
- LO-5.2.9 Reconocer o justificar los criterios de salida adecuados para niveles de prueba específicos y grupos de casos de prueba (p.ej., para las pruebas de integración, las pruebas de aceptación o los casos de prueba para las pruebas de usabilidad). (K2)

Esta sección, Planificación y Estimación de Pruebas, cubrirá los siguientes conceptos clave:

- Niveles y objetivos de la planificación de las pruebas.
- Propósito y contenido del plan de pruebas.
- Estrategias de pruebas.
- Cronogramas de pruebas y priorización de los casos de prueba.
- Planificación de las pruebas para un proyecto, los niveles y las metas.
- Planificación para la preparación y ejecución de las pruebas.

- Criterios de salida de las pruebas.
- Estimación de las pruebas utilizando métricas y experticia.
- Factores de la estimación de las pruebas.

Un plan de pruebas es un plan de proyecto para las actividades de las pruebas, que comienza con la planificación misma y yendo todo el camino hasta las actividades del cierre de las pruebas.

Pensamos que escribir y actualizar los planes de pruebas tienen dos beneficios principales:

En primer lugar, obligándonos a estudiar detenidamente y articular lo que debe ocurrir, nosotros enfrentamos los desafíos, concretamos los pensamientos y nos adaptamos a las circunstancias que cambian.

En segundo lugar, porque escribimos el plan, tenemos una herramienta de comunicación, algo que podemos utilizar para explicar el plan a los probadores, los colegas y los jefes.

Algunas veces, tenemos que escribir varios planes para un único proyecto. Esto pasa cuando estamos abordando dos niveles de pruebas, que ocurrirán en diferentes períodos de tiempo. Esto aplica también a las pruebas que utilizan metodologías y herramientas diferentes (así como las de rendimiento y funcionalidad). Hemos utilizado dos planes diferentes para manejar los esfuerzos de pruebas los cuales mientras eran paralelos en el tiempo, tenían objetivos muy diferentes (p.ej. la prueba de sistema y la prueba beta). También utilizamos dos planes diferentes de pruebas cuando las audiencias eran diferentes, así como con los planes de pruebas del hardware y los planes de pruebas del software.

En este caso, hay un riesgo de superposición y redundancia. Usted puede utilizar entonces un plan maestro de pruebas para describir los elementos comunes así como también asegurar que no hay vacíos o superposiciones inapropiadas entre los esfuerzos.

Porque queremos tener una herramienta de comunicación, lo encontramos útil de hacer circular uno o dos borradores durante la elaboración del plan. Esto promueve la retroalimentación y el debate con anticipación. También previene la pérdida de tiempo si está en el camino incorrecto.

¿Qué debería estar haciendo con su mente durante la planificación de las pruebas?

En primer lugar, debería definir y seleccionar el método y la estrategia de las pruebas y darse cuenta de los niveles de pruebas que tienen que ser realizados. También tendrá que integrar, coordinar y alinear las pruebas con el ciclo de vida del desarrollo del software

Como con cualquier plan, necesita darse cuenta del quién, el qué, el cuándo y el cómo de las actividades de las pruebas.

Para estas actividades y tareas de las pruebas, necesitará asignar recursos.

Basado en los factores abordados en el capítulo 4, necesitará definir la documentación de las pruebas, tanto en cuanto a las plantillas como las herramientas así como también el nivel de detalle o precisión. Recuerde poner el nivel de detalle para los casos y procedimientos de prueba con el fin de proveer suficiente información para apoyar la preparación y ejecución reproducible de las pruebas, si fuera necesario, pero evite la situación del "gorila afeitado" que mencionamos en el capítulo 4.

Y, para que pueda seguir cumpliendo su trabajo durante la ejecución e informar sus resultados, debe seleccionar el monitoreo, el control, las métricas, los diagramas y los informes de las pruebas.

Glosario del ISTQB

Monitoreo de las pruebas: Una tarea de la gestión de pruebas que se ocupa de las actividades relacionadas con la comprobación periódica del estado de un proyecto de pruebas. Informes preparados que comparan los estados reales del proyecto con los planificados. Véase también gestión de pruebas.

Como lo dijimos antes, un plan de pruebas es un plan de subproyecto para la parte de las pruebas de un proyecto. Más adelante, lo guiaremos a través de una plantilla para un plan de pruebas. Ésta es la plantilla IEEE 829.

Usted puede adaptar el esquema IEEE 829 para el uso por cada plan de pruebas en detalle, eso es, el plan de pruebas para cada nivel o fase. Esto también puede servir para el plan maestro de pruebas. Por supuesto, si no tiene como requisito que debe utilizar la plantilla IEEE 829, entonces también puede crear su propia plantilla o esquema.

Recuerde que no debería—en verdad, no puede—escribir el plan de pruebas en un vacío. La planificación de las pruebas y el plan de pruebas serán influenciados por la política de las pruebas en la organización, el alcance de las pruebas, los objetivos de las pruebas y del proyecto, los riesgos del proyecto y la calidad, las características, el cronograma y las restricciones de presupuesto, las áreas de criticidad, la comprobabilidad del sistema y la disponibilidad de los recursos. Idealmente, el mismo plan es también una herramienta para influenciar estas mismas cosas.

Las secciones del plan de pruebas IEEE 829 incluyen lo siguiente:

- Identificador del plan de pruebas.

- Introducción.
- Ítems de prueba (es decir, lo que es entregado para las pruebas).
- Características que deben ser probadas.
- Características que no deben ser probadas.
- Método (las estrategias, la organización y la extensión de las pruebas).
- Criterios paso/falla del ítem (es más raro, para nosotros, porque no pensamos que los ítems pasan o fallan, pero más bien que las características o los atributos no pasan o fallan).
- Criterios de prueba (p.ej., las entradas, las salidas, la suspensión y la reanudación).
- Entregables de pruebas (p.ej., los informes, los diagramas, etc.).
- Tareas de las pruebas (o mínimo los hitos clave).
- Necesidades del entorno.
- Responsabilidades.
- Dotación de personal y necesidades de capacitación.
- Cronograma.
- Riesgos y contingencias (la calidad [del producto] y los riesgos del proyecto).
- Aprobaciones.

Particularmente las secciones esenciales de un plan de pruebas se relacionan con los criterios de las pruebas.

Los criterios de entrada miden si nosotros y el sistema estamos listos para una fase en particular de pruebas. ¿Están los entregables (el objeto de prueba, los ítems de prueba) listos? ¿Está el laboratorio de pruebas listo, incluyendo los casos, los datos, los entornos, las herramientas, etc.? ¿Están listos los desarrolladores, los probadores y otros participantes para comenzar las pruebas?

Los criterios de entrada tienden a ser cada vez más rigurosos mientras las fases avanzan. Es decir, los criterios de entrada para la prueba de sistema tienden a ser más formales que para las pruebas de unidad.

La prueba de sistema puede comenzar cuando:

1. Los sistemas de seguimiento de defectos y seguimiento de pruebas estén instalados.
2. Todas las componentes estén formalmente bajo la configuración y el control automatizado de la gestión de versiones.
3. El equipo de operaciones haya configurado el entorno del servidor para la prueba del sistema, incluyendo todas las componentes y subsistemas del hardware meta. El equipo de pruebas tenga el acceso adecuado a estos sistemas.
4. Los equipos de desarrollo hayan terminado todas las características y las correcciones planificadas de los defectos para la versión.
5. Los equipos de desarrollo hayan realizado las pruebas de unidad de todas las características y las correcciones planificadas de los defectos para la versión.
6. Haya menos de 50 defectos abiertos para la versión, que deben ser corregidos según Ventas, Marketing y Servicio al Cliente.

7. Los equipos de desarrollo hayan proporcionado el software al equipo de pruebas 3 días hábiles antes de comenzar la prueba de sistema.

8. El equipo de pruebas haya realizado 3 días de “pruebas de humo” e informe acerca de los resultados en la reunión inicial de la fase prueba de sistema.

9. El equipo de gestión del proyecto haya acordado de continuar en la Reunión acerca de la Entrada a la Fase Prueba de Sistema. Y los siguientes puntos hayan sido resueltos en la reunión:

- ✓ Si el código está terminado.
- ✓ Si las pruebas de unidad están terminadas.
- ✓ Asignación de una fecha meta para la corrección de cualquier defecto conocido que debe ser corregido (no más tarde que una semana después de la Entrada a la Fase Prueba de Sistema).

Figura 5.5: Ejemplos de Criterios de Entrada

Esta figura muestra un ejemplo de los criterios de entrada de una prueba de sistema para el proyecto del dispositivo de Internet que mencionamos anteriormente.

Pase un momento leyendo estos criterios.

Los criterios del 1 al 3 se refieren a la preparación de las pruebas, al sistema que debe ser probado y al entorno de las pruebas.

Los criterios del 4 al 5 son reglas del modelo del ciclo de vida del desarrollo del software asociadas con la utilización de un modelo del ciclo de vida secuencial, como lo hicimos en este proyecto.

Los criterios del 6 hasta el 8 tienen que ver con la preparación de la calidad; Es decir, ¿Es el sistema que debe ser probado lo suficientemente estable para obtener el beneficio de las pruebas en este momento?

El criterio 9 proporciona el consentimiento positivo de la dirección del proyecto y la organización para comenzar las pruebas.

Otro conjunto de los criterios se refieren a la capacidad de continuar las pruebas. Los criterios de continuación miden si las pruebas pueden continuar de forma eficiente y eficaz. Ellos buscan típicamente los problemas del entorno de las pruebas, los defectos que bloquean las pruebas en el sistema sometido a pruebas o serias fallas del proceso.

Ahora, tenga en cuenta de que “los criterios de continuación” son sólo una manera cortés de decir “criterios de terminación” o “criterios de suspensión/reanudación” al revés. Lo comenzamos a utilizar porque nos seguimos metiendo en problemas en la reunión de revisión del plan de pruebas cuando la gente llegaba a la sección de los criterios de suspensión/reanudación.

Como sea que lo denomine, como regla general, el cronograma es rey en muchos proyectos. De este modo, si invoca a estos criterios para detener las pruebas, es muy poco probable de que se haga popular.

La Prueba de Sistema continuará si:

1. Todo el software publicado para el Equipo de Pruebas está acompañado de las Notas de la Versión.
2. Ninguna modificación ha sido realizada en el sistema, ni en el código fuente, ni en los archivos de configuración, o en otras instrucciones de instalación o en los procesos, sin el acompañamiento de un informe de defecto. Si un cambio es realizado sin un informe de defecto, entonces el Jefe de Pruebas abrirá un informe de defecto urgente solicitando información y lo escalará a su jefe.
3. El retraso de los defectos abiertos (“el vacío de la calidad”) sigue siendo menos de 50. Los períodos de cierre diarios y escalonados siguen siendo menos de 14 días (en promedio, los defectos son corregidos dentro de dos ciclos de versión semanales).
4. Las reuniones de revisión de defectos ocurren 2 veces por semana hasta La Salida de la Fase de Prueba de Sistema para gestionar el retraso de los defectos abiertos y los tiempos de cierre de defectos.

Figura 5.6: Ejemplos de Criterios de Continuación

Este ejemplo muestra los criterios de continuación de la prueba de sistema para el proyecto del dispositivo de Internet.

Pase un momento leyendo estos criterios.

En el criterio 1, pedimos las notas de entrega para cada compilación.

En el criterio 2, pedimos la adherencia a las buenas reglas de gestión de cambios y la adherencia a las reglas secuenciales del modelo del ciclo de vida del Desarrollo de Software que aplican a un modelo secuencial del ciclo de vida.

En el criterio 3 y 4, queremos la gestión dirigida por la gestión positiva de los defectos y la calidad global.

Finalmente veamos los criterios de salida. Los criterios de salida miden si la fase de pruebas puede ser considerada completa. Estos tienden a incluir medidas de rigurosidad, así como la cobertura del código, la funcionalidad o los riesgos. ¿Por qué debemos tener confianza en la aptitud del sistema para nosotros?

Los criterios de salida también pueden involucrar las estimaciones de la densidad de los defectos o las medidas de fiabilidad. ¿Cuántos defectos creemos que quedan? ¿Con qué frecuencia fallará el producto en producción o cuando el cliente lo está utilizando?

Glosario del ISTQB

Densidad de Defectos: El número de defectos identificados en un componente o sistema dividido por el tamaño del componente o sistema (expresados en términos de medidas estándar, p.ej. líneas de código, número de clases o puntos de función).

Los criterios de salida pueden—y probablemente deben—considerar los factores del costo. ¿Cuánto hemos gastado en las pruebas? ¿Cuánto vale el producto? ¿Cuál es el costo comparativo para encontrar el próximo defecto en las pruebas comparado con la producción o el uso del cliente?

Los criterios de salida pueden y deben definitivamente considerar el nivel de los riesgos residuales. ¿Cuáles son los defectos conocidos no corregidos? ¿Cuáles áreas de riesgo no han sido cubiertas? ¿Cuáles riesgos estamos aceptando si terminamos las pruebas en este momento?

Finalmente, todos los criterios de salida, explícitamente o implícitamente, incluyendo las consideraciones de la planificación.

Es importante recordar que, en última instancia, todas las decisiones hechas acerca de los criterios son decisiones de negocios. Si está en el mundo de la seguridad crítica y misión crítica, hay un

motivo por el cual usted puede **negarse** absolutamente a renunciar uno o más criterios. Sin embargo, en la mayoría de las situaciones, los jefes de proyecto y negocios no sólo tienen derecho, pero sí un deber, un deber de responsabilidad mutua, para considerar la calidad en el contexto de la característica, el cronograma y las restricciones de presupuesto, los requisitos legales e industriales, las presiones competitivas y una serie de otros factores del mundo real.

Durante un proyecto, presentamos nuestros descubrimientos de las pruebas en una reunión de revisión del proyecto. Los criterios de salida no fueron alcanzados. Recomendamos continuar las pruebas y la corrección de los defectos.

Un ejecutivo en la reunión dijo, "Bueno, todo es verdad, Rex, pero estamos entregando la versión del producto a la producción al final de esta semana de todas maneras".

Y Nosotros respondimos, ¿"Podríamos saber por qué?"

"Seguro", él dijo. "Me reuní ayer con los inversionistas que fomentan este producto. Ellos dijeron que teníamos que generar ingresos este año o ellos cortarían la inversión. Porque estamos vendiendo un sistema electrónico de consumo, tenemos que ser parte de las ventas de navidad. Eso significa que tenemos que estar en los estantes un día después del Día de Gracias. Eso nos da tres semanas, incluyendo el aumento de la línea de producción para el hardware. No podemos esperar más tiempo que el viernes. Hagan los últimos ajustes en el producto ahora y envíenlo".

Éste es un argumento poderoso, viajando hacia mucho tiempo atrás al sabio que escribió el libro de Eclesiastés, quién dijo, "Mejor un perro vivo que un león muerto". ¿Es la calidad de un producto anulado mejor o peor que la calidad de un producto algo imperfecto que es comprado por miles o millones? ¿Cuántos defectos de un software o sistema justifican poner a una empresa fuera del negocio? Depende del producto y la empresa, ¿eh?

La Prueba de Sistema terminará cuando:

1. No hayan ocurrido cambios (de diseño/código/características), excepto para abordar los defectos de La Prueba de Sistema, que ocurrieron en las 3 semanas anteriores.
2. No haya ocurrido pánico, caída, paro, conflicto de recursos, suspensión del procesamiento, terminación inesperada de un proceso, u otro paro de proceso en algún software de servidor o hardware en las últimas 3 semanas .
3. Ninguno de los sistemas cliente se hayan vuelto inoperables debido a una actualización que falló durante la Prueba de Sistema.
4. El Equipo de Pruebas haya ejecutado todas las pruebas planificadas contra el software candidato de disponibilidad general.
5. Los Equipos de Desarrollo hayan resuelto todos los defectos que debían ser corregidos según Ventas, Marketing y Servicio al Cliente.
6. El Equipo de Pruebas haya comprobado que todas las cuestiones en el sistema de seguimiento de defectos estén ya sea cerrados o postergados, y , donde sea apropiado, sean verificados por las pruebas de regresión y confirmación.
7. Las métricas de las pruebas indiquen: la estabilidad y la fiabilidad del producto; el cumplimiento de todas las pruebas planificadas; la cobertura adecuada de los riesgos de calidad crítica.
8. El Equipo de la Gestión del Proyecto acuerde que el producto, como se lo ha definido durante el ciclo final de la Prueba de Sistema, satisfará las expectativas razonables de calidad del cliente.
9. El Equipo de Gestión del Proyecto tenga una Reunión de Salida de la Fase de Prueba de Sistema y esté de acuerdo que hemos terminado la Prueba de Sistema.

Figura 5.7: Ejemplos de Criterios de Salida

Finalmente, este otro ejemplo muestra los criterios de salida de la prueba de sistema para el proyecto del dispositivo de Internet.

El criterio 1 exige la estabilidad del código antes del lanzamiento, así que lo que hemos probado y lo que producimos son más o menos la misma cosa.

Los criterios 2 y 3 se refieren a los tipos particulares de defectos de que, si son observados, ponen en duda el diseño del sistema.

Los criterios desde el 4 hasta el 6 se refieren a la completitud de la ejecución de las pruebas y la gestión del ciclo de vida de las incidencias para todas las pruebas e incidencias.

El criterio 7 se refiere a las métricas acerca de las pruebas, las cuales veremos en una sección más adelante.

Los criterios 8 y 9, nuevamente, proporcionan una aprobación positiva del proyecto y la gestión organizacional para finalizar las pruebas y entregar el producto. Una vez más, estos proporcionan también un medio por el cual la gestión puede renunciar a los criterios o no tomarlos en cuenta, lo cuál puede ser necesario, dado el tiempo, el cronograma, o las restricciones de las características.

Veamos el desarrollo de una estructura detallada del trabajo ("WBS") para las pruebas.

Recuerde que la estructura detallada del trabajo (“WBS”) es una división jerárquica del trabajo que debe ser realizado en un proyecto. ¿Entonces, cuáles son las principales etapas en su proyecto de pruebas? Una estructura que hemos utilizado es la consideración del proyecto como que consiste de alguna secuencia superpuesta de:

- Planificación.
- Asignación de personal (si aplicable).
- Adquisición y configuración del entorno de las pruebas.
- Desarrollo de las pruebas.
- Ejecución de las pruebas.

Ahora, en cada etapa, necesitamos identificar las actividades principales y luego refinarnas en tareas discretas.

Una manera de identificar las principales actividades es por medio de “una visión hacia el futuro”. Aquí, revisamos mentalmente un proyecto típico de pruebas, preguntándonos a nosotros mismos, basados en nuestra experiencia lo que tiende a ocurrir en cada etapa.

Otra manera de identificar las principales actividades es por medio de “una visión hacia el pasado”. Si utiliza pruebas basadas en los riesgos, lo que hace es comenzar con la ejecución de las pruebas y preguntarse, “¿Cuáles juegos de pruebas son necesarios para los riesgos críticos?” Por ejemplo, podría haber identificado los principales riesgos en las áreas de funcionalidad, rendimiento, control de errores, y así sucesivamente.

Ahora, pregúntese, ¿Cuáles tareas de desarrollo son necesarias para ejecutar estas pruebas? ¿Cuáles tareas del entorno son necesarias para ejecutar estas pruebas? ¿Cuáles tareas de la asignación del personal son necesarias para ejecutar estas pruebas? ¿Cuáles tareas de la planificación son necesarias para ejecutar estas pruebas?

Recuerde, cuando se crea una estructura detallada del trabajo (“WBS”) esas tareas deben ser cortas en duración (p.ej. pocos días). Adicionalmente, los indicadores del progreso claramente visibles—a menudo llamados en inglés “inch-pebbles” que significa hitos menores—deberían ser producidos frecuentemente, en vez de confiar en unos pocos “hitos”. Con cronogramas no detallados e hitos poco frecuentes, puede quedaratrás y no darse cuenta hasta muy tarde en el juego.

¿Qué se debería involucrar en la estimación acerca de cuánto tiempo y cuánto esfuerzo necesitaremos para cada tarea en una Estructura Detallada del Trabajo (“WBS”)?

Hay dos métodos generales para la estimación (incluyendo la estimación de las pruebas). Primero, podemos estimar las tareas individuales por medio del trabajo con el dueño de las tareas o con los expertos. Ésta es realizada de abajo hacia arriba y es lo que hacemos cuando construimos una estructura detallada del trabajo (“WBS”).

Segundo, podemos estimar el esfuerzo de las pruebas basados en las métricas de los proyectos anteriores o similares o basados en los valores típicos. Si aplicamos esto en un nivel más alto, como con una regla heurística para la proporción de probador-a-desarrollador, entonces estamos realizando una estimación descendente. Si utilizamos los parámetros para estimar las actividades individuales como el hallazgo de los defectos, la corrección de los defectos, el desarrollo de los casos de prueba, etc., entonces estamos realizando una estimación ascendente.

Ambos son útiles. Preferimos recurrir a la sabiduría del equipo para crear una estructura detallada del trabajo (“WBS”), y luego aplicar modelos de reglas heurísticas para comprobar y ajustar la estimación. Lo encontramos usualmente más exacto y mucho más defendible contra las tentativas de reducir el cronograma de las pruebas (estas tentativas son posibles en todas partes).

Cuando se estima una tarea de pruebas, tome en cuenta que probar es complejo. El esfuerzo y la duración son influenciados por un número de factores, los cuales caen dentro de las siguientes categorías principales.

Hay factores de los procesos: las pruebas dominantes a través del ciclo de vida, la gestión del control de los cambios, el desarrollo en general y la madurez del proceso de pruebas, el ciclo de vida elegido, la alineación del desarrollo y los procesos de pruebas, los resultados de las fases anteriores de las pruebas, los niveles estimados y reales, y el tiempo necesario para corregir esos defectos.

Hay factores materiales: la disponibilidad de las herramientas, la calidad del sistema de pruebas, la disponibilidad de los entornos de depuración y pruebas separados, la disponibilidad de una buena documentación del proyecto, y la similitud de este proyecto con proyectos anteriores (permitiendo la reutilización).

Hay factores personales: las habilidades del equipo, las expectativas de los participantes, los interesados del negocio y los jefes, el grado de apoyo de los que no están participando, especialmente los patrocinadores, y las relaciones dentro y fuera del proyecto.

Hay factores que hacen demorar: un alto nivel de complejidad, muchos interesados del negocio o en contra de los interesados de los negocios, la demasiada novedad, la distribución geográfica y la zona horaria, la necesidad para producir documentación detallada de las pruebas, las logísticas

dificiles para el ítem de pruebas, los datos de prueba o los productos de los resultados de las prueba, y los datos frágiles de prueba.

Cuando usted está estimando, sólo tiene que entender las técnicas de estimación pero también cómo estos factores influencian la estimación. Las pruebas no existen en el vacío, sino que más bien están presentes en cada parte del proyecto. Así, la desviación de la estimación de las pruebas puede surgir a raíz de factores externos y eventos antes o durante las pruebas.

Cambiemos la marcha ahora. En el capítulo 4, mencionamos 2 ejemplos de las estrategias de pruebas, las analíticas y las reactivas. Explicamos cómo esas dos eran muy diferentes. Hablemos más acerca de las estrategias de pruebas y el método de pruebas.

Glosario del ISTQB

Método de prueba: La implementación de la estrategia de pruebas para un proyecto específico. Típicamente incluye las decisiones tomadas que sigue, basadas en las metas del proyecto (de pruebas) y en la evaluación de los riesgos que son llevados a cabo, los puntos de partida con relación al proceso de pruebas, las técnicas de diseño de pruebas que deben ser aplicadas, los criterios de salida y los tipos de pruebas que deben ser realizados.

La estrategia de pruebas es la forma en la cual un equipo de pruebas procederá en las pruebas, en un sentido general e independiente del proyecto. En algunos casos, las organizaciones han escrito las estrategias de pruebas, y en estos casos el documento de la estrategia de pruebas describirá los métodos de pruebas de la organización. Esto incluye cómo seleccionar, diseñar, implementar y priorizar la secuencia y ejecutar los casos de prueba. Incluye la gestión de riesgos del producto y proyecto. Cubre la división de las pruebas en pasos, niveles o fases, así como las pruebas de componente, pruebas de integración, pruebas de sistema y pruebas de aceptación. Da un esquema general de las actividades de alto nivel asociadas con las pruebas en cada uno de estos niveles de pruebas. Recuerde que estamos hablando acerca de las afirmaciones y las prácticas generales, independientes del proyecto.

Debería recordar que el plan de pruebas IEEE 829 incluye una sección llamada Método de Prueba. El método de prueba es la implementación de la estrategia de pruebas para un proyecto específico. El método de prueba es refinado en la especificación del diseño de pruebas, acerca de lo cual usted se debería recordar del capítulo 4. Entonces, por ejemplo, el plan de pruebas podría decir que nuestro método de prueba para las pruebas del rendimiento debe utilizar una herramienta de pruebas automatizadas del rendimiento. La especificación del diseño de pruebas para el juego de pruebas del rendimiento especificaría entonces la herramienta exacta que estaríamos utilizando, cuál versión de la herramienta que utilizaríamos, nuestras fuentes para los datos de prueba, los niveles de carga a los que someteríamos el sistema, y así sucesivamente.

El método de prueba incluye típicamente las decisiones tomadas acerca de las pruebas, basadas en las metas de las pruebas, las metas del proyecto y la evaluación de los riesgos.

Éste es el punto de partida para la planificación del proceso de pruebas, la selección de las técnicas del diseño de pruebas y los tipos de pruebas que deben ser aplicados y para la definición de los criterios de entrada y salida.

Diferentes situaciones invocan diferentes métodos para las pruebas. Por ejemplo, si un equipo de pruebas es involucrado muy tarde en el proyecto, con tiempo mínimo de preparación, será forzado a confiar intensamente en un método reactivo. En situaciones donde las regulaciones requieren la cobertura completa de los requisitos con las pruebas, ese método no funcionaría por sí mismo; un método analítico basado en los requisitos debe ser aplicado, ya sea sólo o de acuerdo con otros métodos. Entonces el método que debe ser seleccionado depende del contexto. Cuando se selecciona un método, considere los riesgos, los daños y la seguridad, los recursos y las habilidades disponibles y la tecnología.

Describamos las estrategias de pruebas comúnmente utilizadas.

Primeramente, encontramos las estrategias analíticas. Los dos ejemplos más comunes son la estrategia analítica basada en los riesgos—la estrategia por defecto explicada en este libro—y la analítica basada en los requisitos.

Seguidamente, tenemos las estrategias basadas en el modelo. En estas estrategias, construimos uno o más modelos del sistema en los cuales se apoyan para nuestras pruebas. Por ejemplo, en las pruebas de los sistemas telefónicos, los modelos del crecimiento de la fiabilidad—los modelos matemáticos prediciendo la fiabilidad del campo basados en la tasa de llegada de incidencias durante las pruebas de carga—son utilizados para preparar, ejecutar y evaluar la completitud de las pruebas.

Algo similar a las estrategias analíticas son las estrategias metódicas con respecto a la apariencia. Aquí, seguimos una metodología pre-planificada, p.ej., basada en las fallas, basada en listas de comprobación o basada en la calidad. La diferencia entre éstas y el método analítico es que con frecuencia no hay una fase del análisis donde las listas de comprobación o la taxonomía de los defectos sean adaptadas al proyecto, como sería en el método analítico.

Seguidamente vienen las estrategias en base a procesos o compatibles con un estándar. Por ejemplo, hemos visto que algunos clientes toman la clase de SQE y el libro, Pruebas de Software Sistématicas, junto con el estándar IEEE 829 del cuál David Gelperin desarrolló ese curso, y

construyó un proceso de pruebas completo alrededor de eso. Al otro lado del formalismo las metodologías Ágiles como el desarrollo dirigido por pruebas ("TDD") — las cuales, a propósito, no quieren decir que los probadores dirigen a los desarrolladores o el desarrollo. Note que cuando usted se apoya en un proceso desarrollado externamente y/o estándar para su estrategia, usted tiene la gran esperanza, de que la gente que desarrolló ese proceso y estándar, tenían su situación particular ¡en mente!

Seguidamente, como anteriormente fue explicado, tenemos también las estrategias dinámicas. Éstas se caracterizan por un método más reactivo a los eventos y las condiciones que las pre-planificadas o las pre-diseñadas, junto con la fuerte dependencia de las técnicas basadas en la experiencia.

Para aquellas personas que no pueden o no decidirán qué probar, les podemos ofrecer las estrategias consultivas o dirigidas. Esto es una manera refinada de decir que exigiremos mucho de los que no prueban, así como los desarrolladores, los analistas de negocio y otros, para que nos digan qué probar. La diferencia entre ésta y las estrategias analíticas basadas en los riesgos es que, en las pruebas basadas en los riesgos, nos apoyamos en todos los interesados del negocio para que nos proporcionen información y actúen como corresponsables en el proceso. En las estrategias dirigidas, usualmente los probadores piden información solamente a un interesado de negocios o a un grupo de interesados del negocio, y luego aceptan esa información como una verdad revelada. ¡Sólo funciona cuando usted es dirigido por una persona realmente inteligente!

Finalmente, tenemos las estrategias de pruebas contrarias a la regresión. Aquí, la cosa más importante es asegurar que la funcionalidad existente no sea dañada. Estas estrategias se apoyan fuertemente en la reutilización del material de las pruebas, la automatización extensiva de las pruebas y los juegos estándar de pruebas. Por supuesto, estas estrategias no funcionarán bien en los productos de crecimiento rápido.

5.2.1 Ejercicios

Ejercicio 1

Esquematice un plan de pruebas para Omninet.

Utilizando la plantilla IEEE 829 como un punto de partida, escriba de dos a cinco ítems o sentencias de alto nivel en cada sección.

Argumente.

Solución del Ejercicio 1

Identificador del Plan de Pruebas

OSTP-001

Introducción

- Introducir el proyecto Omninet.
- Mencionar que las pruebas cubren el quiosco, el centro de llamadas y el centro de datos.
- Esquematizar los riesgos de calidad claves del análisis de los riesgos.

Ítems de pruebas (es decir, lo que es entregado para probar)

- Hacer una lista del hardware y el software del quiosco.
- Hacer una lista del software del centro de llamadas (el hardware es comercial y no hecho a la medida).
- Hacer una lista del software del centro de datos (el hardware es comercial y no hecho a la medida).

Características que deben ser probadas

- Hacer una lista de las áreas de los riesgos de calidad para las cuales las pruebas ocurrirán.

Características que no deben ser probadas

- Hacer una lista de las áreas de los riesgos de calidad para las cuales las pruebas no ocurrirán, para aquellas áreas las cuales la gente podría pensar que usted cubrirá.
- Mencionar que las pruebas de unidad e integración serán cubiertas en otra parte.

Método (estrategias, organización, grado de cobertura de las pruebas)

- Describir la estrategia de pruebas. Nuestras estrategias para Omninet incluirían las pruebas basadas en el riesgo, pruebas automatizadas de carga o rendimiento y regresión y pruebas mezcladas de guiones o exploratorias.
- Mostrar el organigrama para las pruebas (véase el ejercicio anterior).

- Describir los varios grados de cobertura de las pruebas y hacer referencia al análisis de los riesgos de calidad.

Criterios de paso o falla de los ítems

- Explicar en un alto nivel cuáles criterios de paso o falla se aplicarán.
- Explicar cuáles oráculos de pruebas serán utilizados (p.ej. diferentes computadoras de escritorio y portátiles para comparar el comportamiento del quiosco).
- Describir cómo las pruebas automatizadas comprobarán los resultados.

Criterios de pruebas (p.ej. entrada, salida, suspensión y reanudación)

- Hacer una lista de los criterios de entrada (cuando esté listo para comenzar a probar).
- Hacer una lista de los criterios de salida (cuando esté listo para terminar las pruebas, y, por inferencia porque ésta es una prueba de sistema, cuando el sistema pueda ser enviado).
- Hacer una lista de cualquiera de las situaciones que causarían que las pruebas sean detenidas temporalmente.

Entregables de pruebas (p.ej. informes, diagramas, etc.)

- Describir los informes clave.
- Describir el seguimiento interno de los resultados de las pruebas.
- Describir los casos de prueba, las herramientas y los datos que serán entregados para las pruebas de mantenimiento.

Tareas de pruebas (por lo menos los hitos clave)

- Hacer una lista de los hitos clave de la estructura detallada del trabajo.
- Asegúrese de incluir la finalización del análisis de los riesgos de calidad, el plan de pruebas y la configuración del entorno de pruebas en el lado de las pruebas.
- Asegúrese de incluir la finalización del plan del proyecto, el plan de la gestión de configuración, las construcciones tempranas y las que están listas para ser probadas en el lado del desarrollo.

Necesidades del entorno

- Describir el entorno de las pruebas, incluyendo los quioscos, el centro de llamadas y el centro de datos.
- Porque ésta es una primera versión, ¿Puede ser utilizado el entorno de la producción? Si es así, ¿Cuándo debería ocurrir la creación del entorno de las pruebas de mantenimiento y podríamos medir ese entorno contra el de producción para realizar comparaciones posteriores de los resultados de las pruebas con el comportamiento esperado en producción?

Responsabilidades

- Describir los participantes clave de las pruebas y lo que ellos hacen.
- Describir los participantes clave que no son de las pruebas, quienes se relacionan con las pruebas (p.ej. el jefe de desarrollo, el ingeniero de versiones, etc.) y lo que ellos hacen.
- Dibujar un diagrama de configuración mostrando estos participantes y los entregables entre ellos.

Necesidades de la asignación de personal y la capacitación

- Describir a cualquier persona que debe ser contratada.
- Describir la capacitación necesitada para el equipo existente (p.ej. en herramientas automatizadas, etc.).

Cronograma

- Hacer referencia a la estructura detallada del trabajo.

Riesgos y contingencias (calidad [producto] y riesgos del proyecto)

- Hacer referencia al documento del análisis de los riesgos de calidad.
- Hacer una lista de los riesgos claves de proyecto que podrían afectar las pruebas. Dar acciones de contingencia o mitigación para cada uno.

Aprobaciones

- Aprobaciones por los revisores e interesados del negocio claves en las pruebas.

5.3 Monitoreo y Control del Progreso de las Pruebas

Objetivos del Aprendizaje

LO-5.3.1 Recordar las métricas usuales e utilizadas para el monitoreo de la preparación y la ejecución de las pruebas. (K1)

LO-5.3.2 Explicar y comparar las métricas de las pruebas para los informes de las pruebas y el control de las pruebas (p. ej. Los defectos encontrados y corregidos, y las pruebas pasadas y falladas) relacionados con el propósito y el uso. (K2)

LO-5.3.3 Resumir el propósito y el contenido del documento del informe del resumen de las pruebas de acuerdo al "Estándar para la Documentación de las Pruebas de Software" (Estándar IEEE 829-1998). (K2)

Glosario del ISTQB

Control de pruebas: Una tarea de la gestión de pruebas que se encarga del desarrollo y la aplicación de un conjunto de acciones correctivas para encaminar un proyecto de pruebas cuando el monitoreo muestra una desviación de lo que fue planificado. Véase también gestión de pruebas.

Esta sección, Monitoreo y Control del Progreso de las Pruebas, cubrirá los siguientes conceptos clave:

- Métricas comunes utilizadas para el monitoreo de la preparación y la ejecución de las pruebas.
- Comprender e interpretar las métricas de las pruebas.
- Propósito y Contenido del documento del informe de las pruebas según el estándar IEEE 829.

Glosario del ISTQB

Informe de pruebas: Consulte el informe del resumen de las pruebas.

Informe del resumen de las pruebas: Un documento que resume las actividades y los resultados de las pruebas. Éste contiene también una evaluación de los ítems de pruebas correspondientes con los criterios de salida.

Hay una serie de métricas de las pruebas bastante comunes. Éstas incluyen lo siguiente:

- El porcentaje de realización de la preparación de los casos de prueba (el porcentaje de los casos de prueba preparados y planificados).
- El porcentaje de realización de la preparación del entorno de pruebas (éste es potencialmente engañoso, si cuenta el número de productos instalados o que algunos productos son más difíciles que otros).
- El cómputo y los porcentajes de la ejecución de los casos de prueba. Por ejemplo, el número de casos de prueba ejecutados/no ejecutados y el número de casos de prueba pasados/fallados. Nuevamente, esto puede ser engañoso cuando estos números son utilizados como una métrica sustituta para la calidad, porque una o dos pruebas pueden fallar con errores críticos y pueden hacer que el producto no pueda ser enviado.
- Varias clases de diagramas y métricas de defectos, así como la densidad de los defectos, los defectos encontrados y corregidos y los resultados de la repetición de pruebas.
- El alcance de la cobertura de las bases de pruebas, así como los requisitos, los riesgos o el código, por medio de las pruebas. Algunas veces esto puede incluir los resultados de paso/falla.
- Otro es el nivel de confianza, usualmente un número subjetivo, que los probadores tienen en el producto. Usted está contando con su credibilidad si elige utilizar éste.
- Las fechas de los hitos claves de pruebas, incluyendo alguno en el pasado que podría o no podría haber sido encontrado a tiempo.

Finalmente, los costos de las pruebas, incluyendo el costo del hallazgo del próximo defecto o la ejecución de la próxima prueba comparada con el beneficio. Recuerde que esto vincula a los criterios de salida que mencionamos antes en una sección anterior.

Estas métricas pueden ser utilizadas para los informes de las pruebas. El informe de las pruebas puede ser acerca del resumen o el análisis de los resultados de las pruebas.

Los análisis de las métricas pueden ocurrir en los eventos clave así como la medición del cumplimiento de los criterios de salida para una reunión propuesta acerca de la salida de las pruebas.

También podemos analizar varias métricas para realizar recomendaciones o guiar los proyectos. Esto puede incluir la consideración de las métricas como el número estimado de los defectos restantes, los costos y los beneficios de más pruebas, el nivel residual de los riesgos o el nivel subjetivo de confianza.

Para evitar la gestión de nuestras pruebas completamente por medio de la intuición, nos gusta utilizar métricas. Estas métricas pueden ayudarnos a evaluar si los objetivos de las pruebas fueron adecuados para el nivel de pruebas que estamos abordando. Esto puede también ayudarnos a evaluar la adecuación de las estrategias y actividades de pruebas. Esto puede también ayudarnos a medir la efectividad y la eficiencia de nuestras pruebas, basadas en nuestros objetivos.

Las métricas también pueden ser utilizadas para el control de las pruebas. Podemos seleccionar y planificar para las acciones correctivas y las que sirven como guía basadas en la información y las

métricas de las pruebas. Esto puede influenciar las actividades de las pruebas—u otras actividades del ciclo de vida del software.

Algunos ejemplos de las acciones del control de las pruebas incluyen lo siguiente:

Si durante las pruebas exploratorias, encontramos un gran número de defectos importantes en un área evaluado como de bajo riesgo, entonces realizamos la redefinición de las prioridades de las pruebas y la redistribución del esfuerzo de las pruebas.

Si la fecha de la disponibilidad planificada para un entorno de pruebas cambia, entonces ajustamos el cronograma de las pruebas.

Si durante las pruebas de confirmación, observamos una alta tasa de informes de defectos reabiertos, entonces la gestión del proyecto adiciona un nuevo criterio de entrada con la necesidad de la repetición de las pruebas de las correcciones de los defectos por los desarrolladores antes de la integración en una construcción.

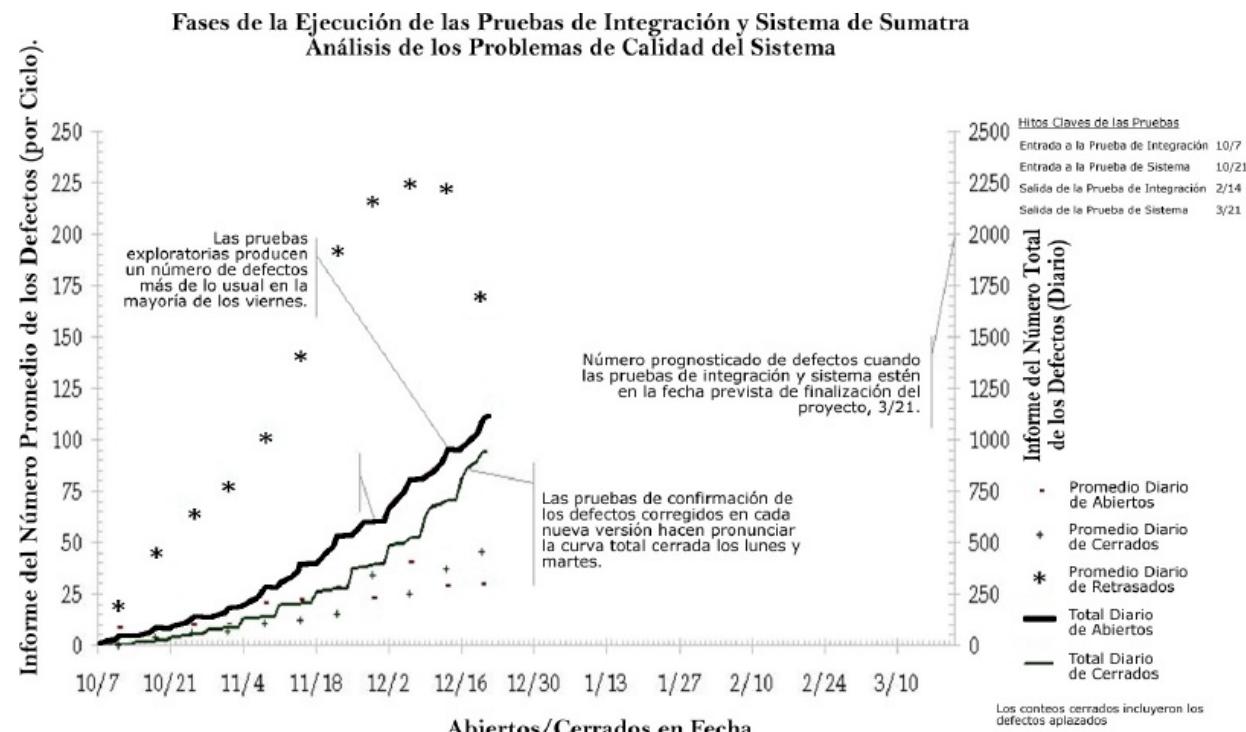


Figura 5.8: Ejemplo de métricas de pruebas

Veamos algunos ejemplos de las métricas de pruebas. Todas estas métricas se derivan del mismo proyecto, Sumatra30, por eso usted puede observar similitudes y diferencias. Comencemos con un ejemplo del diagrama de la tendencia de los defectos comunes.

Este diagrama tiene cinco conjuntos de datos principales. Dos de los cuales están trazados en el eje de la derecha.

El primer conjunto de datos, mostrado con la línea gruesa sólida, es el número acumulativo de los defectos encontrados. Esto está representado en la gráfica contra el eje de la derecha. El segundo, mostrado con la línea delgada sólida, es el número acumulativo de los defectos resueltos, también esto está representado en la gráfica contra el eje de la derecha.

Eso incluye ambos, los defectos corregidos y los defectos pospuestos, porque el interés de este diagrama es demostrar si el estado de los defectos está tiendiendo hacia la preparación para la entrega.

El tercer conjunto de datos, mostrado con los símbolos “menos”, es el número promedio de los defectos abiertos en una semana calendario. Esto está representado en la gráfica contra el eje de la izquierda. El cuarto, mostrado con los símbolos “más”, es el número promedio de los defectos resueltos en una semana dada de calendario, también graficado contra el eje izquierdo.

El quinto conjunto de datos, mostrado con los símbolos “asterisco”, es el atraso promedio en una semana de calendario dada.

Desde el momento que los diagramas de tendencias muestran las tendencias en el tiempo, no es casualidad, que este diagrama incluye notaciones para explicar varias formas de las curvas, así como la nivelación de la línea acumulativa del hallazgo de los defectos durante el feriado de acción de gracias. Sin esas anotaciones, los observadores pueden malinterpretar los ecos del proceso como una señal que estamos tratando de interpretar, en este caso la calidad del producto.

Fases de la Ejecución de las Pruebas de Integración y Sistema de Sumatra

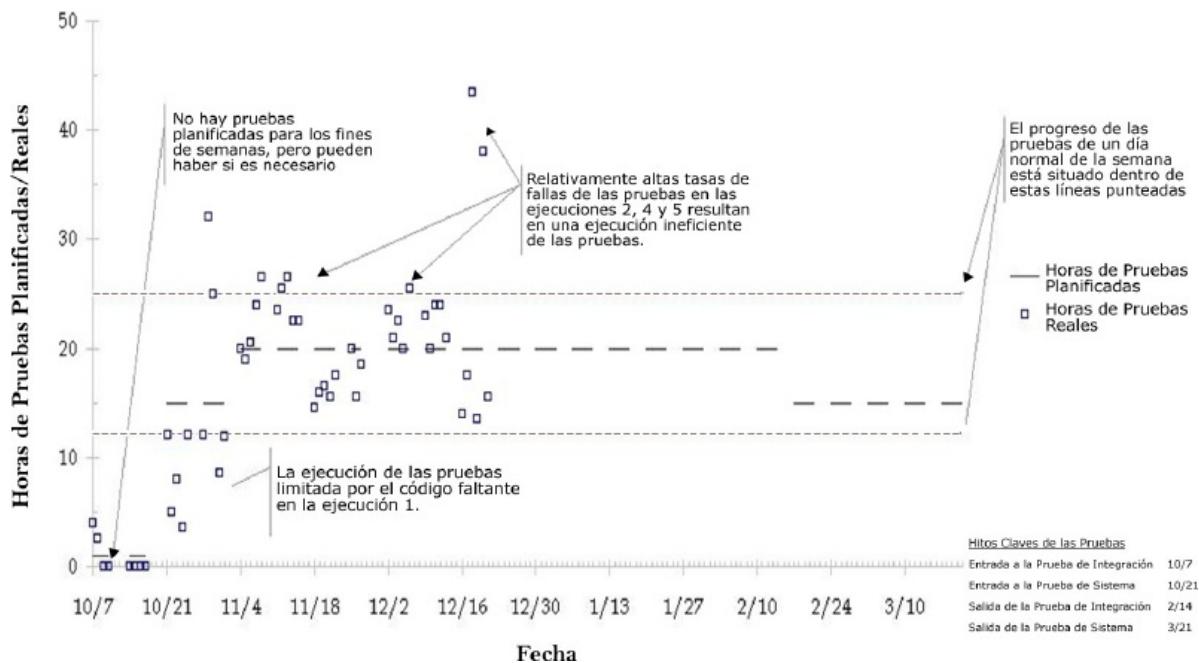


Figura 5.9: Progreso de las pruebas

Otro diagrama típico muestra la tendencia del progreso a través del tiempo.

Este diagrama tiene dos conjuntos de las pruebas principales, ambos trazados en contra del eje único del lado izquierdo. El primer conjunto de los datos son las horas planificadas de la ejecución de las pruebas por día, representadas por las líneas discontinuas. Tome en cuenta que las horas cambian primero hacia arriba, luego hacia abajo, durante el proyecto, indicando una intensidad divergente y planificada de las pruebas.

El segundo conjunto de datos son las horas reales de la ejecución de las pruebas logradas por fecha, las cuales están representadas por las cajitas, donde pensamos que el progreso del día de la semana es normal. En otras palabras, las cajitas pueden saltar alrededor aleatoriamente arriba y abajo de las líneas discontinuas, con la condición de que ellas permanezcan dentro de las líneas punteadas. Para aquellas líneas fuera de las líneas punteadas, hemos proporcionado una explicación del por qué.

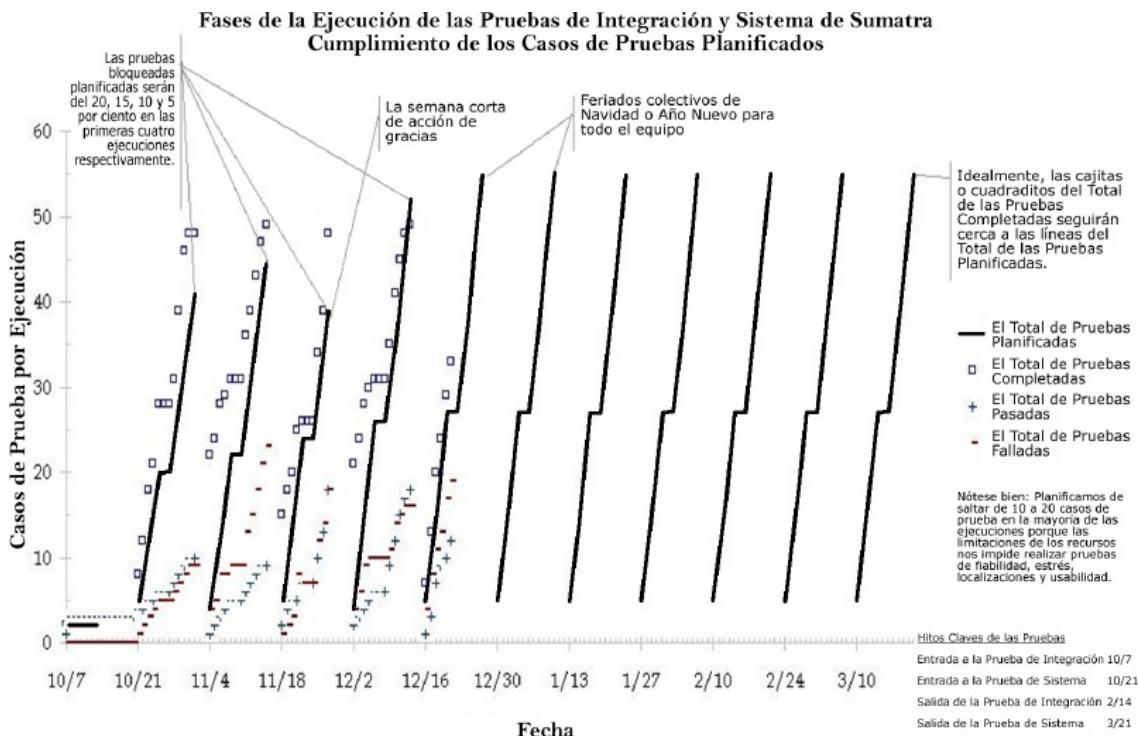


Figura 5.10: Cumplimiento de los casos de pruebas planificados

Otro diagrama típico muestra la tendencia de la realización de los casos de prueba en el tiempo.

Este diagrama tiene cuatro conjuntos de datos principales, todos trazados contra el eje izquierdo. El primer conjunto de datos es el número de las pruebas planificadas que deben ser completadas en un paso dado de las pruebas, mostradas con líneas parecidas a "rayos". Esta sección plana en la mitad de la línea parecida a un rayo ocurre debido a los fines de semana, cuando no se han programado pruebas. El restablecimiento de la realización de las pruebas—en otras palabras, por qué cada línea parecida a un rayo comienza cerca del cero—es debido al hecho de que estamos ejecutando una secuencia de ejecuciones de pruebas similares, utilizando la repetición de pruebas para la gestión de los riesgos de regresión.

El segundo conjunto de datos es el número real de las pruebas completadas para la ejecución en el final de cada día, representado con las cajitas.

El tercer conjunto de datos es el número de aquellas pruebas completadas las cuales han pasado en el final del día, representado con los símbolos "más".

El cuarto conjunto de datos es el número de aquellas pruebas completadas las cuales han fallado en el final del día, representadas con los símbolos "menos".

Nuevamente, observe que hemos utilizado las notaciones para indicar los aspectos interesantes de los conjuntos de datos, especialmente por qué razón ciertas ejecuciones de las pruebas son más cortas que otras.

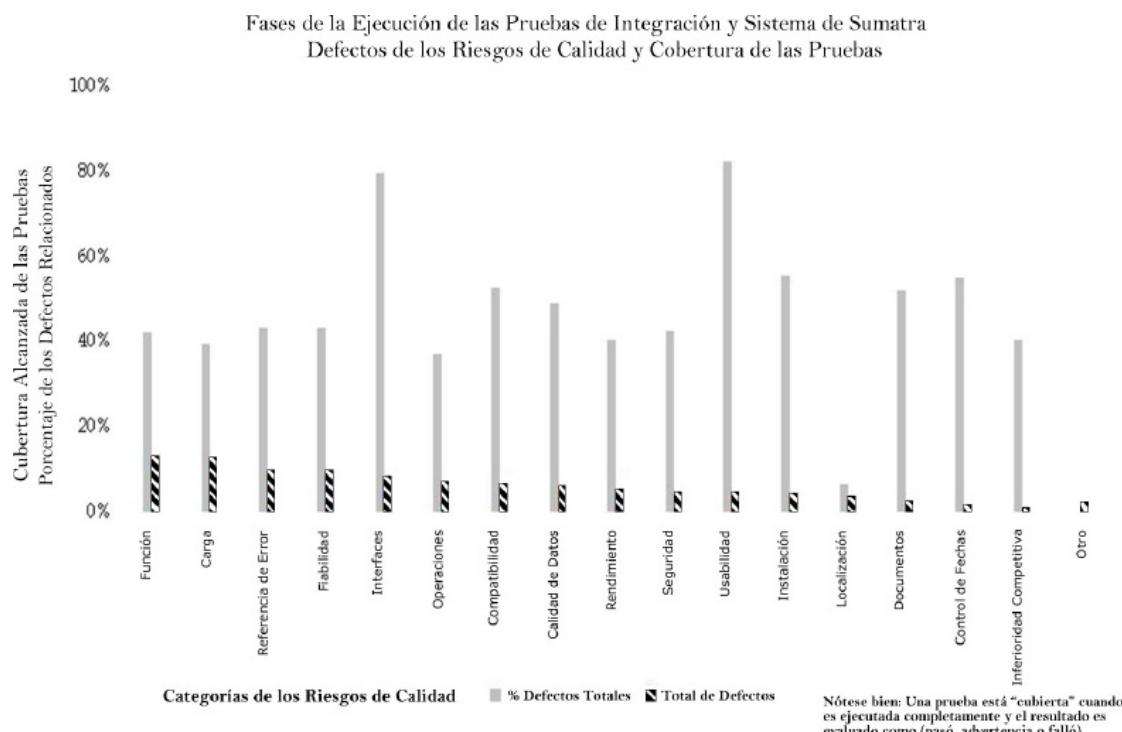


Figura 5.11: Defectos de Riesgos de Calidad y Cobertura de Pruebas

Otro diagrama, no tan típico pero útil, analiza la relación entre las categorías clave de los riesgos de calidad (presumiblemente nuestra base de pruebas), el grado de la cobertura de los riesgos que hemos alcanzado para cada categoría y el número de los defectos que hemos encontrado relacionados a cada categoría de los riesgos.

Abajo, tenemos etiquetas que corresponden a cada categoría principal de riesgo. Tenemos la trazabilidad de nuestros casos de prueba hacia atrás a los riesgos de los cuales se derivaron, como los describimos en el capítulo 4. Utilizamos esa trazabilidad, en el nivel no detallado de la categoría de riesgo, para crear este diagrama. También necesitaremos la clasificación de nuestros informes de los defectos en cuanto a los riesgos con los cuales se relacionan, y también utilizaremos esa trazabilidad para crear este diagrama.

Esta información acerca de la trazabilidad de la cobertura de las pruebas es utilizada para representar gráficamente el porcentaje alcanzado de la cobertura planificada de los riesgos en cada categoría, como está representado por las barras grises. Mientras que las barras grises crecen hacia el 100%, la cantidad de los riesgos residuales en esta categoría bajan, porque hay menos y menos que no hemos probado todavía. En el 100%, hemos alcanzado la mitigación planificada de los riesgos para esta categoría. No existen más cuestiones desconocidas y ni pendientes para explorar y el único riesgo para esta categoría se relaciona con los defectos conocidos. Dese cuenta que no hay pruebas planificadas para "otras", porque la categoría de riesgo es utilizada para clasificar los defectos que encontramos y que no conducen a un riesgo identificado. Esto será útil en la comprobación de nuestro análisis de los riesgos de calidad durante la ejecución de las pruebas, como lo hablamos en el capítulo 4.

La información acerca de la trazabilidad de la cobertura de los defectos es utilizada para representar gráficamente el porcentaje de los defectos relacionados con cada categoría, como está representado con las barras rayadas negras y blancas. Estas barras, si están puestas una sobre la otra, harán siempre el total del 100%. Si la barra de los defectos es alta en comparación con la barra de las pruebas, entonces eso indica un riesgo importante, porque hay otro gran número de defectos ya encontrados, aunque sólo hemos ejecutado pocas pruebas. En cambio, si la barra de los defectos es pequeña cerca a una barra grande de las pruebas, entonces el riesgo es bajo, porque hemos alcanzado una alta cobertura de pruebas y no hay muchos defectos aquí. Si la barra de los defectos arriba la "otra" categoría está sobre el 5%, debemos comenzar a dudar de nuestro análisis de los riesgos de calidad. Si están sobre el 10%—en otras palabras, más que 1 en 10 defectos ocurridos en cualquier área no esperamos encontrar muchos defectos o donde decimos que los defectos no serían importantes—entonces debemos revisar definitivamente nuestro análisis de los riesgos de calidad.

Adicionalmente a la especificación del diseño de pruebas, la especificación de los casos de prueba, la especificación de los procedimientos de prueba y las plantillas del plan de pruebas, el estándar IEEE 829 para la documentación de las pruebas también incluye una plantilla para los informes del resumen de las pruebas.

Un informe del resumen de las pruebas describe los resultados de un nivel dado o fase de pruebas. La plantilla IEEE 829 incluye las siguientes secciones:

- El identificador del informe del resumen de las pruebas.
- Resumen (p.ej., lo que fue probado, lo que las conclusiones son, etc.).
- Varianzas (del plan, de los casos, de los procedimientos).
- Evaluación comprensiva.
- Resumen de resultados (p.ej., métricas finales, cálculos).
- Evaluación (de cada ítem de prueba con relación a los criterios paso/falla).
- Resumen de las actividades (utilización de los recursos, eficiencia, etc.).
- Aprobación.

Los resúmenes pueden ser entregados durante la ejecución de las pruebas, como parte de un informe o recopilación del estado de los proyectos. Ellos también pueden ser utilizados en el final de un nivel de pruebas como parte de las actividades del cierre de pruebas.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
El Resumen de los Casos de Prueba del Servicio / DataRocket																						
Primera Ejecución																						
ID=ID RPN = # de Prioridad de Riesgo																						
T = Prueba ejecutada S = Prueba Saltada P = Prueba Pendiente F = Prueba Falló																						
Pw = Peso de la Prueba Q = Prueba en Cola y Lista para ser Ejecutada P = Prueba en Progreso B = Prueba Bloqueada																						
7	Prueba		Sistema	Defecto	Defecto	Plan.	Real	Plan.	Real	Plan.	Real	Comentario	T	S	P	#	RPN	I				
8	Propietario	ID	Juego de Pruebas / Caso de Pruebas	Estado	Config.	ID	RPN	Per.	Fecha	Ejecutar	Ejecutar											
9	System cockers, Inc.																					
10	2000	Puebas del Entorno																				
11	2001	Perf/ Término de Operaciones																				
12	2002	Temperatura / Ciclo de Humedad de Operaciones																				
13	2003	Temperatura / Ciclo de Humedad de Operaciones																				
14	2004	Corte no operativo																				
15	2005	Secuadno no operativa																				
16	2006	Secuadno térmica no operativa																				
17	2007	Corte de embalaje																				
18	2008	Secuadno de embalaje																				
19		Resumen del Juego de Pruebas																				
20	Wingard Bytes																					
21	LTW	Carga, Capacidad y Volumen																				
22	2001	CPU y Memoria																				
23		Defecto	A,B,C	DOB	3	LTW	7/10	7/6	4	6	6	7										
24	2002	FD/HD/CD-ROM/DVD																				
25	2003	RAID																				
26	2004	Cassette																				
27	2005	Red																				
28	2006	Banco de Modem																				
29	2007	Usr/Firewall/Serial																				
30	2008	Todos los subsistemas																				
31		Resumen del Juego de Pruebas																				
32																						
33	JC	Funcionalidad Básica																				
34	2000	Configuración Región NT																				
35	2001	Configuración Región Soberia																				
36	2002	Configuración Región Novell																				
37	2003	FD/HD/CD-ROM/DV																				
38	2005	RAID																				
39	2006	Cassette																				
40	2007	Red NT																				
41	2008	Red Novell																				
42	2009	PCI-UPS																				
43	2010	Banco de Modem																				
44	2011	USB/Firewall/Serial																				
45	2012	IU/Video/Keyboard																				
46		Resumen del Juego de Pruebas																				
47																						
48	HL	Estándares																				
49	4001	Logo de Solaris																				
50	4002	Logo de Windows NT																				
51	4003	Logo de Novell																				
52		Resumen del Juego de Pruebas																				

Figura 5.12: Resumen de casos de prueba para un nivel de pruebas

En esta figura, puede ver el resumen de los casos de prueba para un nivel de prueba. Es una hoja de cálculo que hace el seguimiento, prueba tras prueba, del estado de las pruebas. Los casos de prueba, agrupados en los juegos de pruebas, son mostrados en la columna C. Por cada juego de pruebas, tenemos un responsable o asignado (en la columna A), la persona o grupo responsable de la ejecución de la prueba. Cada juego de pruebas y caso de prueba es representado con un número de identificación en la columna B.

En la columna D, capturamos el estado de las pruebas después de ejecutarlas. La columna E hace el seguimiento de la prueba hacia atrás del entorno en el cuál fue ejecutada. Si encontramos uno o más defectos mientras se ejecuta una prueba, el número de los informes de defecto y el número de

prioridad del riesgo del informe de defecto son representados en las columnas F y G.

La columna H tiene las iniciales del probador quién ejecutó la prueba.

Las columnas de la I hasta la N son utilizadas para recopilar las métricas relacionadas con la fecha de ejecución real y planificada, el esfuerzo planificado y real y la duración real y planificada. Estas métricas son esenciales si se quiere utilizar la estimación basada en las métricas de datos históricos, como lo describimos en las secciones anteriores de este capítulo.

De las columnas P hasta la W capturan la información del estado y la prioridad de los riesgos para el resumen como está representado en la siguiente figura.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1															
2		Resumen del Juego de Pruebas del Servidor DataRocket													
3		Primera ejecución													
4		Casos	Pruebas Planificadas Realizadas			Fallas	Pruebas Planificadas No Realizadas			Valor Ganado					
5	Juego de pruebas	Totales	Cantidad	Saltadas	Pasadas	Falladas	ponderadas	Cantidad	En Cola	En Progreso	Bloqueadas	Esfuerzo Planificado	Esfuerzo Real	% Esfuerzo	% Ejecutadas
7	Pruebas del Entorno	8	8	0	8	0	0.00	0	0	0	0	22.00	22.00	100%	100%
8	Carga, Capacidad y Volumen	8	8	0	7	1	0.67	0	0	0	0	32.00	34.00	105%	100%
9	Funcionalidad Básica	12	12	0	12	0	0.00	0	0	0	0	18.00	17.00	94%	100%
10	Estándares	3	3	0	3	0	0.04	0	0	0	0	24.00	24.00	100%	100%
11															
12	Total	31	31	0	30	1	0.71	0	0	0	0	96.00	97.00	101%	100%
13	Por Porcentaje		100%	0%	97%	3%		0%	0%	0%	0%				

Figura 5.13: Resumen del juego de pruebas

Aquí está el resumen del juego de pruebas, una descripción de alto nivel de la información recopilada de la figura anterior, la cuál era el resumen de la ejecución de los casos de prueba. Éste da una buena descripción, juego de pruebas tras juego de pruebas, acerca de dónde estamos en cuanto a la realización de las pruebas. Éste utiliza la palabra “realización” en vez de finalización, porque las pruebas, las cuales son saltadas, son contadas como realizadas y no sería bastante exacto de llamar aquellas pruebas como finalizadas o completadas.

La falla ponderada es la suma de las inversas de los números de prioridad de los riesgos para cada defecto encontrado por las pruebas en cada juego de pruebas. Esto es un poco confuso, pero la manera fácil de pensar en esto es la métrica de la capacidad del juego de pruebas de encontrar muchos defectos o defectos importantes o ambos. Mientras el número sube, el juego de pruebas es un buscador de defectos más efectivo.

El valor ganado es el concepto de la gestión del proyecto que dice, esencialmente, el porcentaje de los hitos y las tareas alcanzadas en un proyecto deben ser casi igual a los recursos utilizados. Durante la ejecución de las pruebas, el caso de prueba es la tarea básica. Las horas del esfuerzo de los probadores es el típico recurso que utilizamos para realizar las pruebas, al menos si el esfuerzo de las pruebas es principalmente manual. Entonces, el porcentaje del gasto del esfuerzo planificado de las pruebas debe ser casi igual al porcentaje de los casos de prueba ejecutados.

Usted debería examinar la figura anterior para ver cómo es derivado cada elemento de esta hoja de cálculo.

Acaba de ver un ejemplo de un registro prueba tras prueba. El estándar IEEE 829 para la documentación de las pruebas incluye también sugerencias acerca de qué incluir en un registro de pruebas.

De acuerdo al estándar, un registro de pruebas debe guardar los detalles relevantes acerca de la ejecución de las pruebas. La plantilla estándar incluye las siguientes secciones:

- Identificador del registro de pruebas.
- La descripción de las pruebas, incluyendo los ítems sometidos a prueba (con los números de versión), los entornos de las pruebas utilizados y similares.
- Datos de actividades y eventos. Estos deben ser descritos prueba tras prueba y evento tras evento. Los eventos incluyen cosas como por ejemplo los entornos de las pruebas no están disponibles, las personas están ausentes por enfermedad, y así sucesivamente. Debe capturar la información acerca del proceso de la ejecución de las pruebas, los resultados de las pruebas, cambios o cuestiones del entorno, defectos, incidentes, o anomalías observadas, los probadores involucrados, alguna suspensión o bloqueo de las pruebas, los cambios del plan y el impacto de los cambios y así sucesivamente.

Como ya ha visto, las hojas de cálculo del seguimiento que capturan estos y muchos otros detalles son frecuentemente utilizados para este propósito.

5.3.1 Ejercicios

Ejercicio 1

En este ejercicio le presentaremos tres escenarios describiendo un proyecto con problemas, y luego verá algunos diagramas de monitoreo del progreso de las pruebas ilustrando el estado.

¿Por cada escenario, cómo presentaría estos resultados de las pruebas al equipo del proyecto?

Si está trabajando con un grupo en este libro, argumente su solución con otros en su grupo antes de continuar y ver nuestra solución.

Descripción del proyecto en problemas:

El proyecto es SpeedyWriter, una aplicación web de procesador de texto.

El equipo de desarrollo ejecutó las fases Prueba de Componente y Prueba de Integración. La Prueba de Componente fue ejecutada del 11 hasta el 29 de Diciembre. La Prueba de Integración fue ejecutada del 18 de Diciembre hasta el 12 de Enero.

El proyecto está ahora en la fase Prueba de Sistema. El primer ciclo de las pruebas fue del 8 hasta el 14 de Enero. Hoy es el 15 de Enero. La Reunión acerca de la Salida de la Fase Prueba de Sistema está planificada para el 29 de Enero (después de dos ciclos de pruebas semanales).

En el primer escenario, el equipo de pruebas está trabajando productivamente y encontrando unos pocos más defectos de lo esperado. Además, existe un gran atraso de los defectos (alrededor de 30 informes). Dado la tasa actual de hallazgo de defectos—la cual no está disminuyendo—y el atraso, el plan para finalizar las pruebas en dos semanas está en riesgo. La alta tasa de hallazgo de defectos lo obligó a saltar un gran número de pruebas en el ciclo uno.

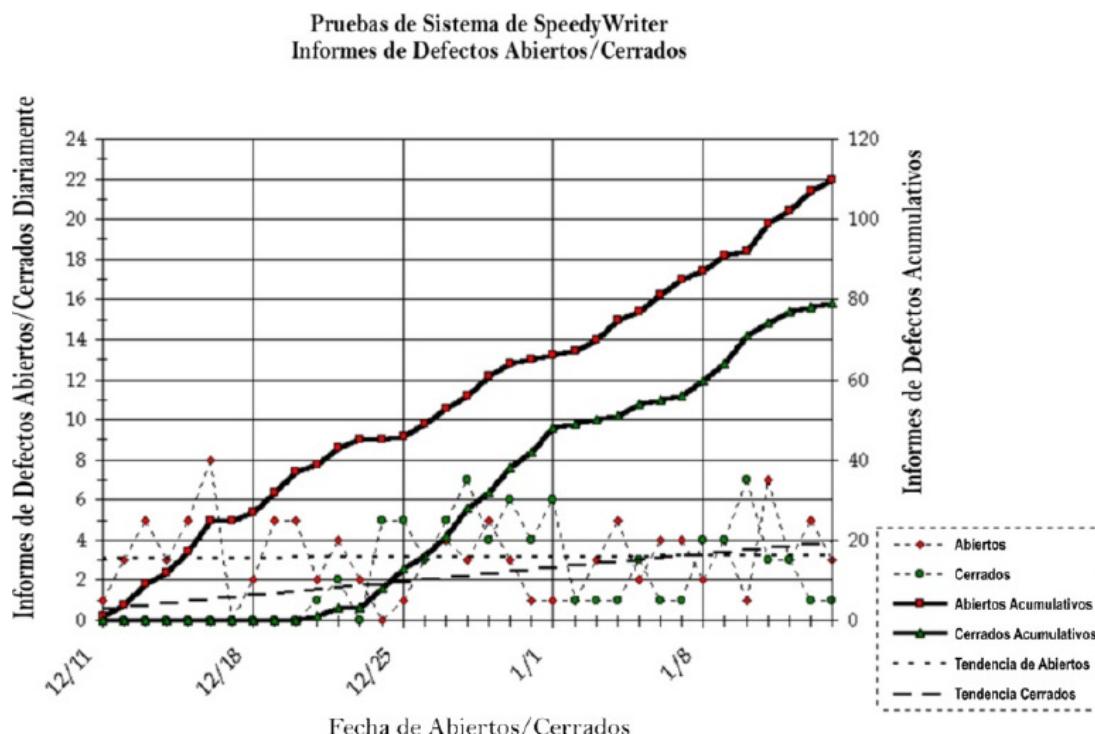


Figura 5.14

Este diagrama muestra los informes de los defectos abiertos y resueltos hasta la fecha en este escenario.

Pruebas de Sistema de SpeedyWriter
Progreso de las Pruebas en Horas

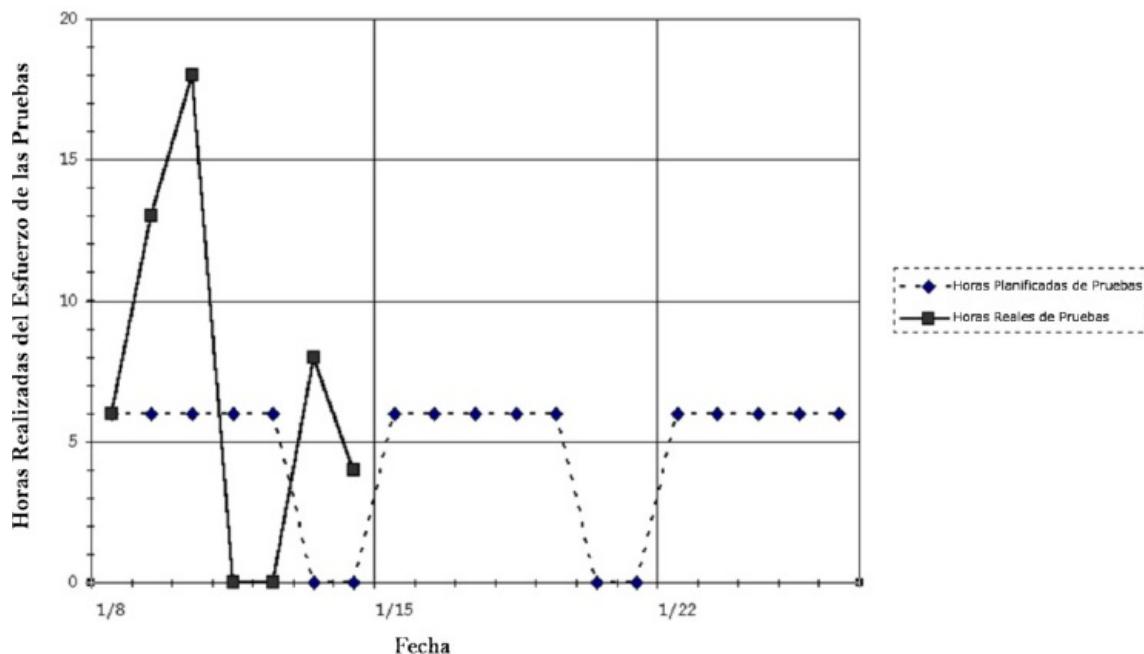


Figura 5.15

Este diagrama muestra las horas de las pruebas realizadas hasta la fecha en este escenario.

Pruebas de Sistema SpeedyWriter
Realización de las Pruebas Planificadas

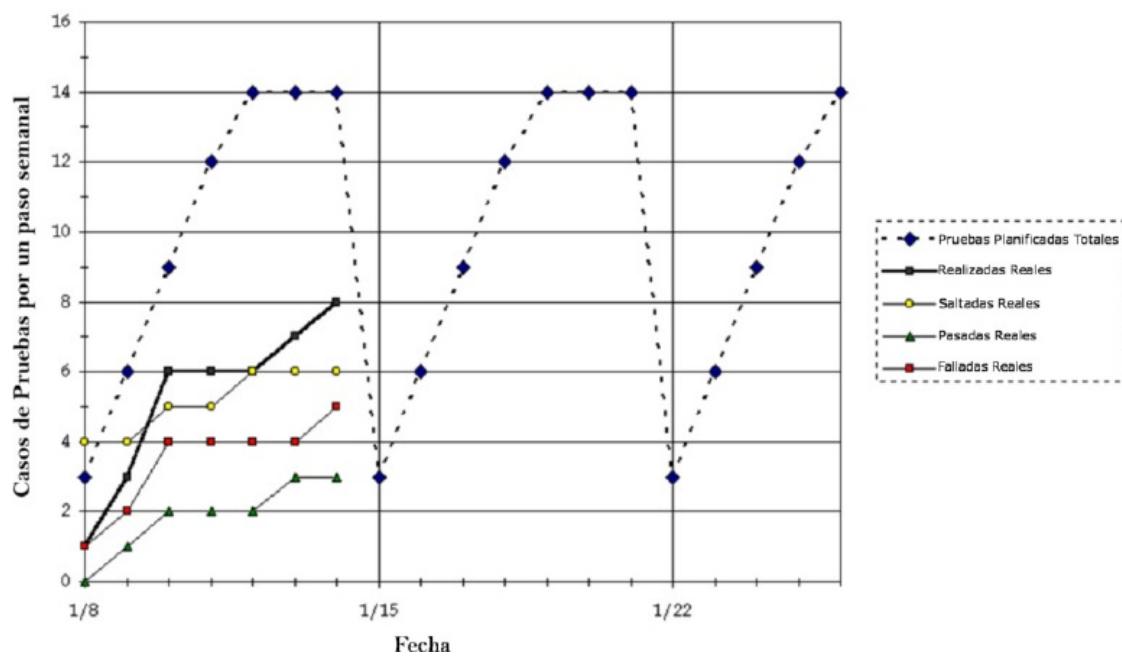


Figura 5.16

Este diagrama muestra los casos de prueba realizados en este escenario hasta la fecha.

Estadísticas de SpeedyWriter
Defectos y Pruebas de la Cobertura de Riesgos de Calidad

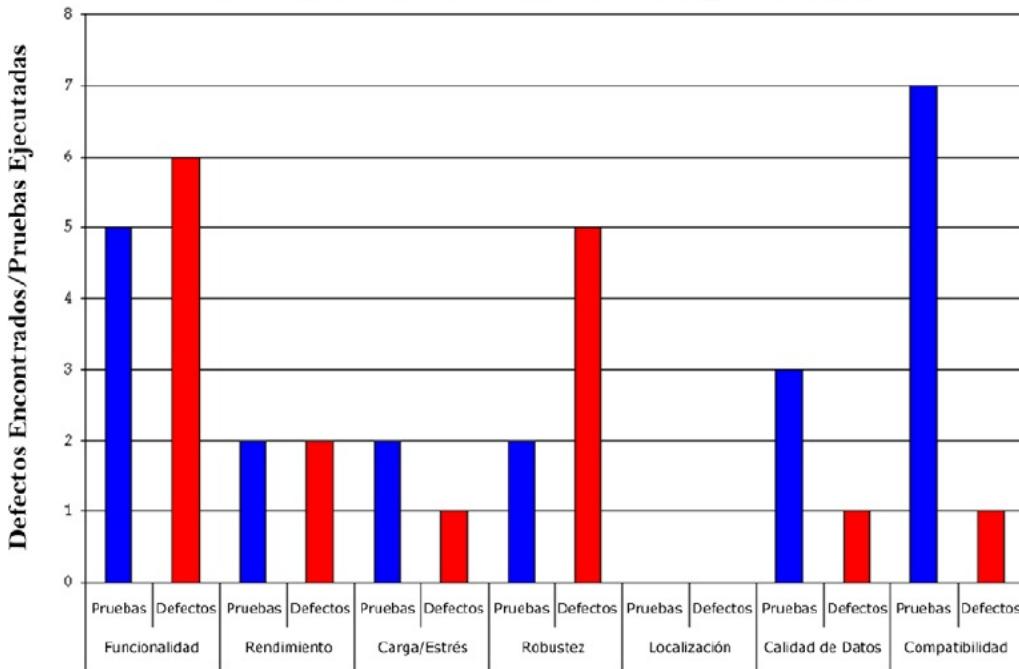


Figura 5.17

Este diagrama muestra el riesgo residual en este escenario hasta la fecha, basado en la cobertura de las pruebas realizadas y los defectos encontrados.

En el segundo escenario, el equipo de pruebas ha sido bloqueado totalmente durante la primera semana de las pruebas. Los equipos de operaciones y de desarrollo no pudieron instalar una construcción que funcione.

El lunes, después de intentar probar todo el día, se determinó que había la versión incorrecta del software en el servidor Web.

El martes, el equipo de operaciones encontró que era necesaria la reinstalación del sistema operativo en el servidor Web.

El miércoles, el equipo de desarrollo produjo cinco construcciones no instalables, una tras otra.

El jueves, a pesar del retoque por el equipo de operaciones, los clientes de las pruebas no pudieron localizar el servidor en la red de pruebas.

El viernes, debido a lo que el equipo de operaciones al final determinó fue un problema de configuración, la aplicación hacía caer al servidor repetidamente cada vez que ésta era puesta en marcha.

El sábado y el domingo, el equipo de pruebas intentó recuperar, pero el browser se caía repetidamente durante las pruebas. Las páginas y las llamadas de teléfono celular no eran devueltas al equipo de desarrollo.

Entonces, las pruebas comenzarían pero luego pararían cuando estos ejercicios tediosos ocurren paso a paso.

Pruebas de Sistema de SpeedyWriter Informes de Defectos Abiertos/Cerrados

**Pruebas de Sistema de SpeedyWriter
Informes de Defectos Abiertos/Cerrados**

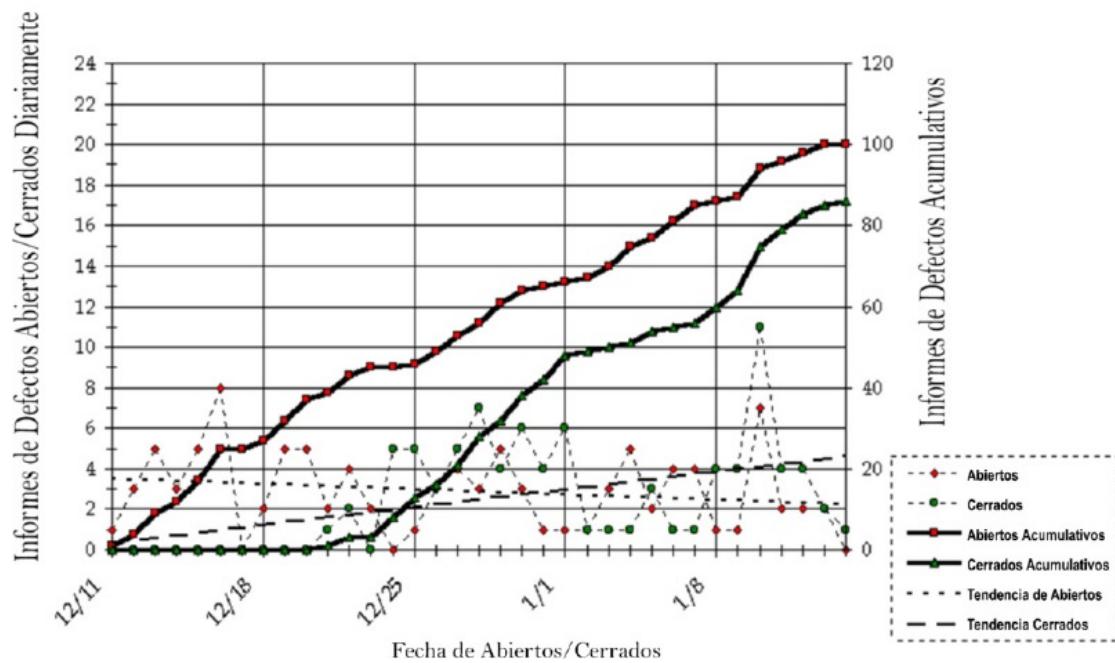


Figura 5.18

Este diagrama muestra los informes de los defectos abiertos y resueltos hasta la fecha en este escenario.

**Pruebas de Sistema de SpeedyWriter
Progreso de las Pruebas en Horas**

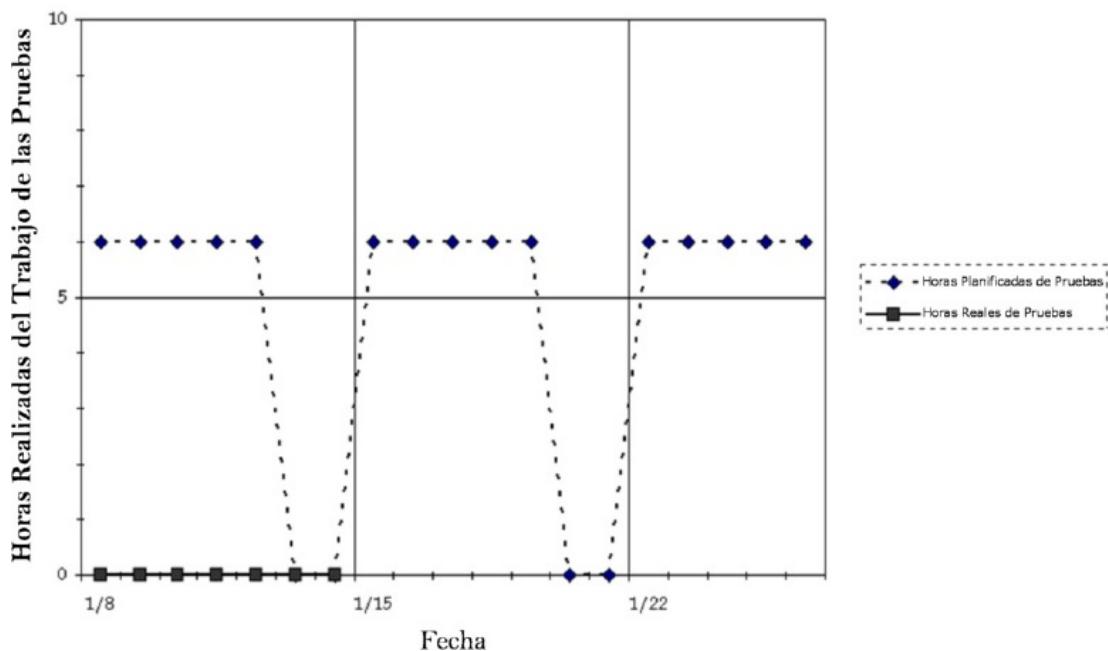


Figura 5.19

Este diagrama muestra las horas de pruebas alcanzadas hasta la fecha en este escenario.

Pruebas de Sistema SpeedyWriter Realización de las Pruebas Planificadas

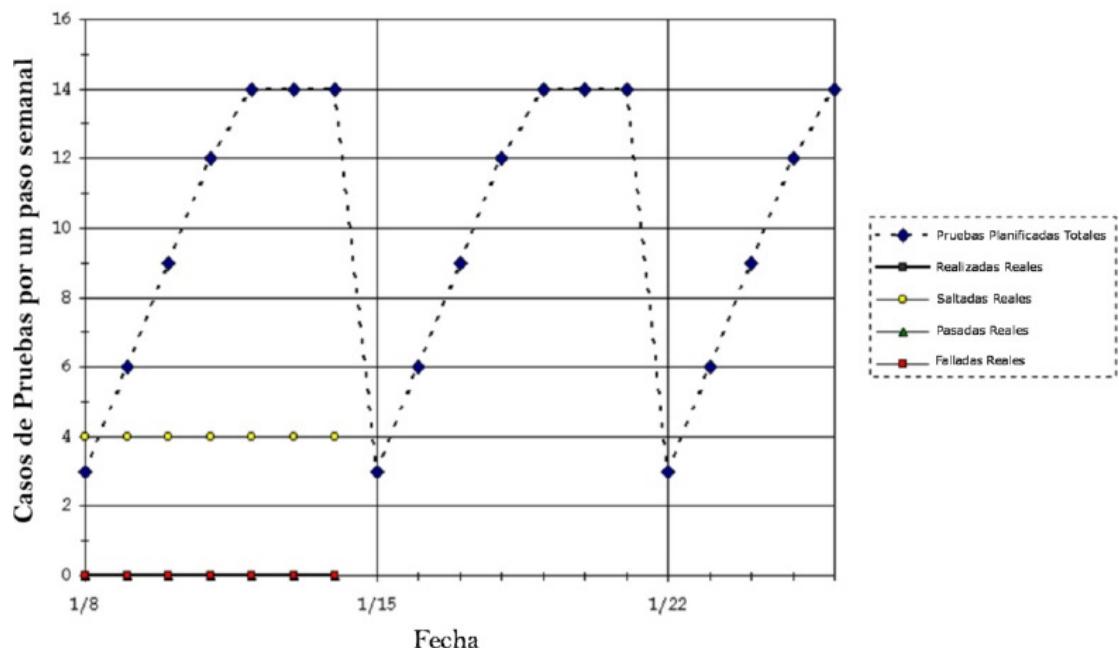


Figura 5.20

Este diagrama muestra los casos de prueba realizados hasta la fecha en este escenario.

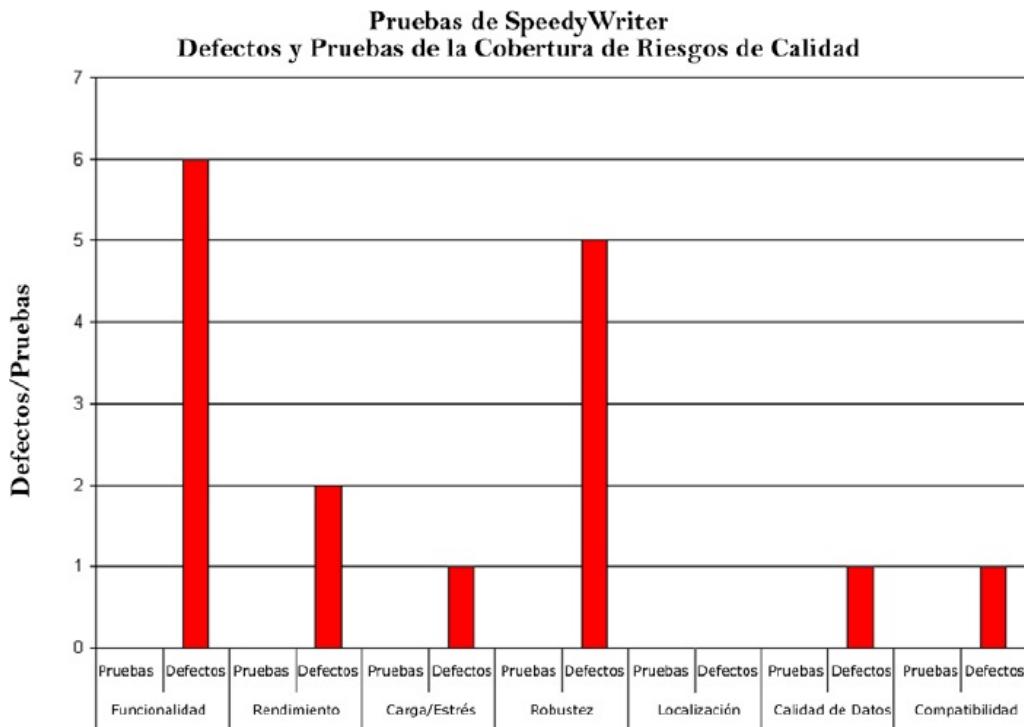


Figura 5.21

Este diagrama muestra el riesgo residual hasta la fecha en este escenario, basado en la cobertura realizada de las pruebas y los defectos encontrados.

En el tercer escenario, el equipo de pruebas encontró muchos defectos en el primer ciclo. Como jefe de pruebas, adicionó probadores adicionales para intentar concordar con el plan. Sin embargo, cada vez que alguien comenzó una prueba, ellos encontraban un defecto tras defecto.

Los problemas son varios y todavía en áreas básicas que simples pruebas habrían capturado. Por ejemplo, la aplicación no puede guardar un documento más largo que una página.

El número y naturaleza de los problemas bloquean muchas pruebas más sofisticadas.

Además, ahora recuerde que el equipo de desarrollo ha estado advirtiendo las pruebas informalmente de que no había tiempo en el cronograma para las pruebas suficientes de componente y de integración.

Pruebas de Sistema de SpeedyWriter Informes de Defectos Abiertos/Cerrados

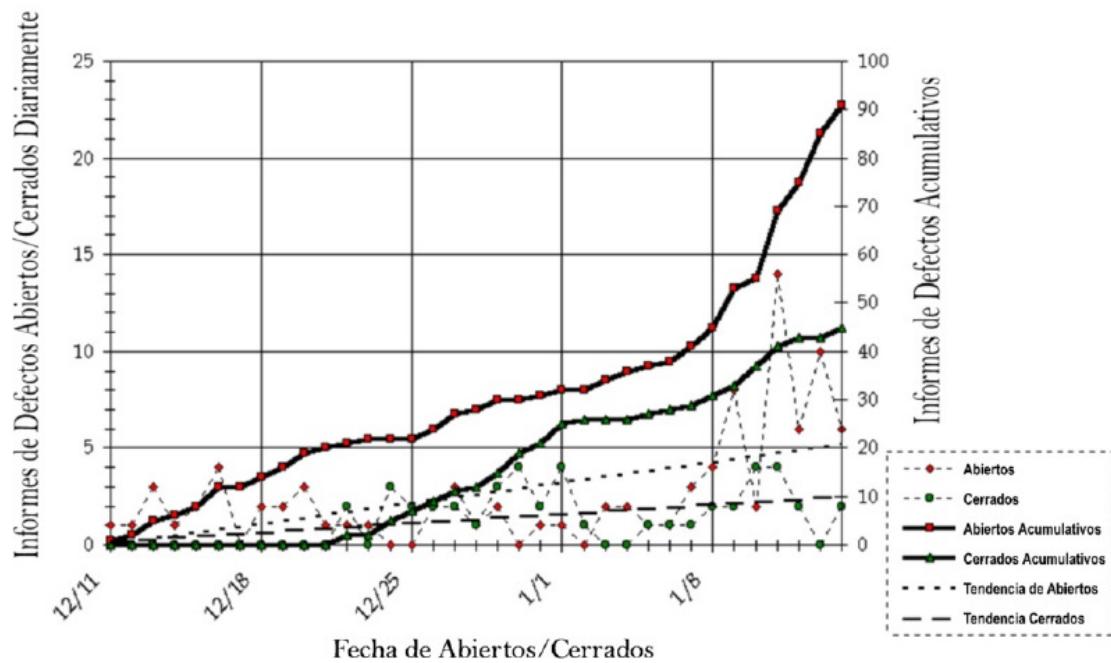


Figura 5.22

Este diagrama muestra los informes de los defectos abiertos y resueltos hasta la fecha en este escenario.

Pruebas de Sistema de SpeedyWriter Progreso de las Pruebas en Horas

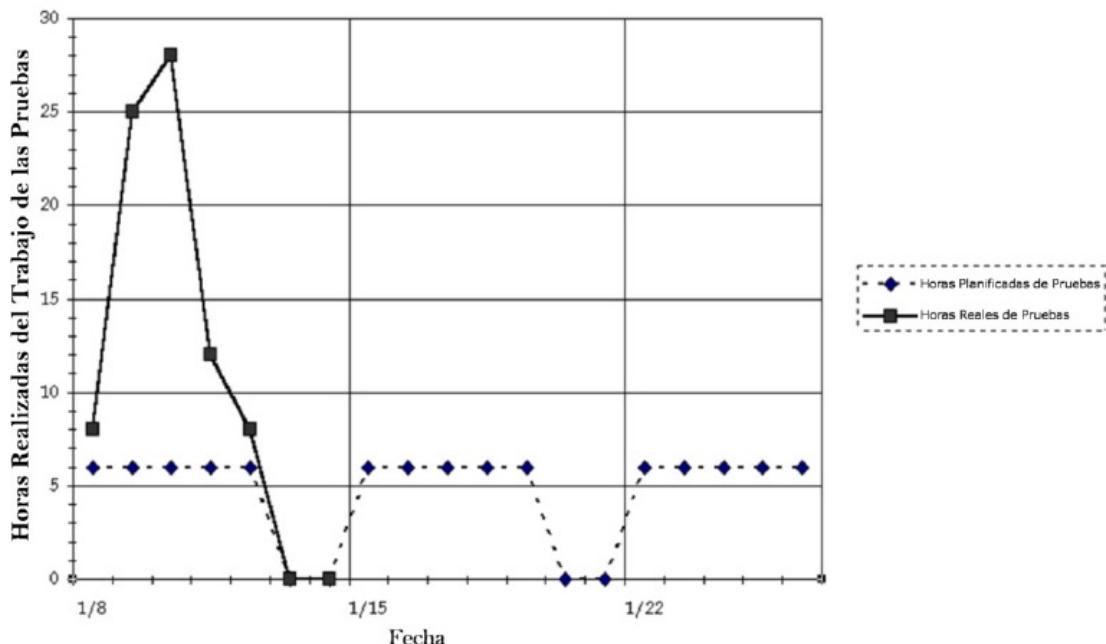


Figura 5.23

Este diagrama muestra las horas de pruebas alcanzadas en este escenario.

Pruebas de Sistema SpeedyWriter
Realización de las Pruebas Planificadas

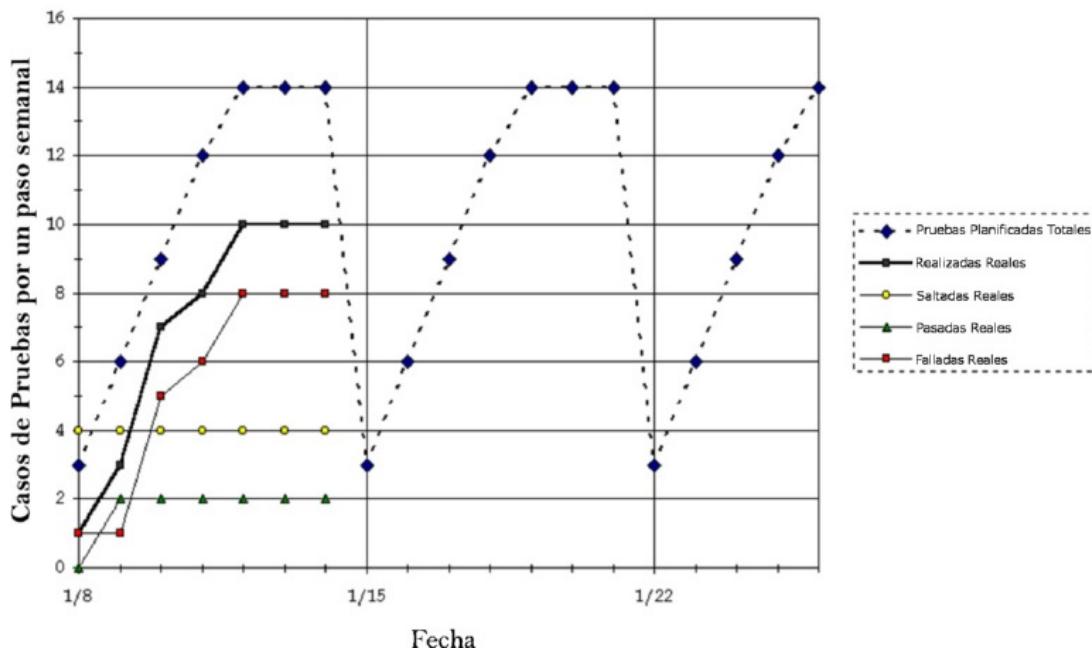


Figura 5.24

Este diagrama muestra los casos de prueba realizados hasta la fecha en este escenario.

Pruebas de SpeedyWriter
Defectos y Pruebas de la Cobertura de Riesgos de Calidad

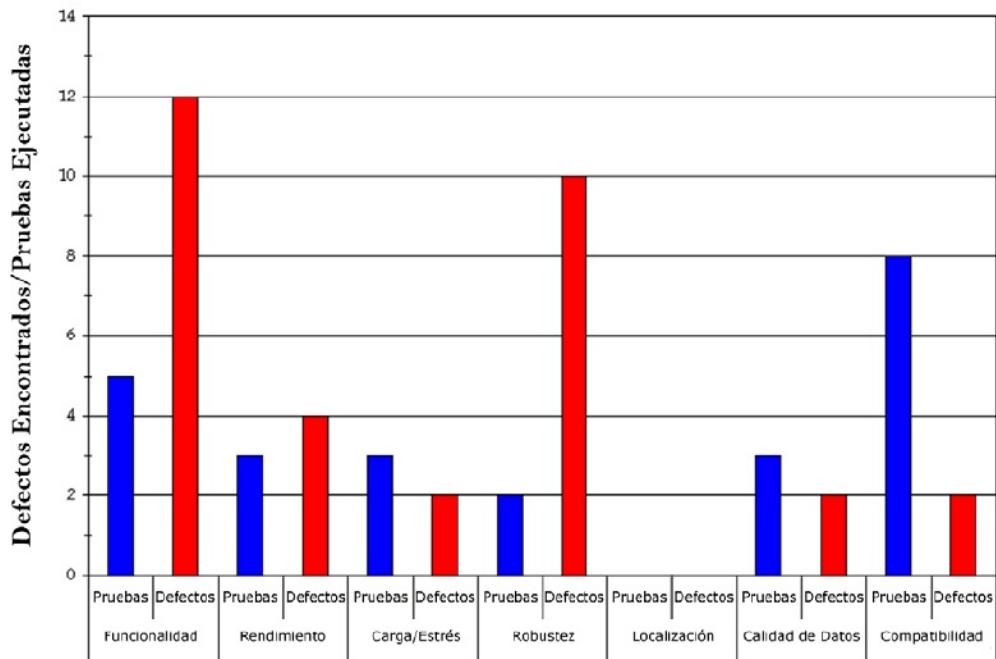


Figura 5.25

Este diagrama muestra el riesgo residual hasta la fecha en este escenario, basado en la cobertura de pruebas realizadas y los defectos encontrados.

Solución del Ejercicio 1

En el primer escenario, el cronograma de las pruebas (y por ende la fecha de la versión) está amenazado por el gran retraso de los defectos y el alta tasa del hallazgo de los defectos. Mencionando algunos de los defectos más significativos, usted puede ayudar a darle a su presentación credibilidad. Sin embargo, asegúrese de no dañar su credibilidad enfatizando demasiado la amenaza; si la mayoría de los defectos son de baja prioridad, asegúrese de mencionar eso.

Habiendo descrito el problema, las posibles soluciones incluyen el retraso de la fecha de la versión, agresivamente clasificando los defectos y priorizando nuevamente las pruebas (para enfocarse en

las áreas más esenciales) y dejando de lado la funcionalidad problemática o aún no probada

En el segundo escenario, el punto principal es no dejar que tal situación se agrave por toda una semana. Comunicar a un superior apropiadamente pero rápidamente. Algunos participantes en una clase dijeron que ellos conseguirían ayuda urgente del personal de desarrollo y de la administración del sistema en el medio día del primer lunes de las pruebas y asegurarían que la gerencia senior se enterara de la situación si ésta persistía en el final del día.

En el tercer escenario, el desafío es el tratamiento de los asuntos políticos asociados con las pruebas inadecuadas de componente e integración y de las indirectas que se le han dado a usted al respecto. Mientras usted pueda que quiera anunciar en la reunión del proyecto, "Escuchen, el jefe de desarrollo metió la pata, incluso su propia gente dijo que él estaba metiendo la pata y ahora no voy limpiar su desastre por él", procediendo así crearía típicamente tanta disensión y enemistad a largo plazo que cualquier satisfacción inicial que podría venir de éste se esfumaría pronto. En cambio, la mayoría de la gente está de acuerdo con ir al jefe de desarrollo temprano en la semana y crear juntos un plan de acción para la recuperación, y anunciando ese plan conjuntamente en la reunión del proyecto sería políticamente el plan más astuto, y, a través de una feliz coincidencia, el plan más probable para conducir el proyecto al éxito.

5.4 Gestión de Configuración

Objetivos del Aprendizaje

LO-5.4.1 Resumir cómo la gestión de configuración apoya a las pruebas. (K2)

Glosario del ISTQB

Gestión de configuración: Una disciplina que aplica la dirección y la vigilancia técnica y administrativa para: identificar y documentar las características funcionales y físicas de un ítem de configuración, controlar los cambios de aquellas características, grabar e informar el estado del proceso de los cambios y de la implementación y verificar la conformidad con los requisitos especificados.

Esta sección, Gestión de configuración, cubrirá los siguientes conceptos clave:

- Cómo la gestión de configuración apoya las pruebas.

La gestión de pruebas y configuración tienen una relación fuerte pero a menudo invisible. Cuando la gestión de configuración es realizada correctamente, nosotros como probadores rara vez lo notamos, pero cuando es realizada incorrectamente, sufrimos realmente.

La gestión de configuración es un conjunto de actividades complicadas y poco valoradas—mucho como en el caso de las pruebas—las cuales se ocupan de la gestión de todos los ítems que componen un componente dado en un sistema, en el sistema completo, y, en los centros de datos y las aplicaciones complejas, en el sistema de sistemas o en la familia de sistemas.

Por ejemplo, usted probablemente utiliza Linux, Windows o Mac en su computador. Cada uno de aquellos paquetes no es solamente un sistema operativo, si no que una familia de sistemas. El sistema operativo mismo consiste en cientos de archivos individuales en su computador. Cada uno de aquellos paquetes contiene también decenas de aplicaciones complementarias como los navegadores, los reproductores de medios, los programas simples de edición, los programas de redes, las herramientas de gestión de sistemas, y similares, cada uno de los cuales a su vez consiste de diez o más archivos.

Cada uno de los archivos en todos estos programas, incluyendo los cientos de archivos en el sistema operativo, tiene que estar construido de decenas o cientos o aún miles de archivos. Cada uno de los archivos fuentes usados para construir los programas y el sistema operativo contiene cinco, diez, o aún cien unidades discretas de código, funciones u objetos dependientes del lenguaje de programación. Cada una de esas unidades discretas de código es potencialmente e individualmente modificable en el transcurso de un proyecto. Cada modificación a una de esas unidades crea una versión distinta de esa unidad. Sólo una versión de cualquier unidad puede ser incluida en un objeto individual de pruebas. La gestión de configuración, realizada correctamente, debe gestionar esa complejidad.

Entonces, la gestión de configuración, realizada correctamente, establecerá y mantendrá la integridad de estos ítems—hasta las unidades individuales—que forman el software, el sistema, el sistema de sistemas o la familia de sistemas en todo el proyecto y ciclo de vida del producto. Para las pruebas, la gestión de configuración debe abordar las siguientes cuestiones:

La gestión del testware y los resultados. Después de todo, los casos de prueba, los datos de prueba, los guiones de prueba y todos los otros documentos similares tienen versiones que están usadas. Tenemos que saber cuál versión del testware fue usada para ejecutar las pruebas, si tenemos que decir por seguro lo que los resultados de las pruebas significan—y lo que ellos no significan.

La gestión de los objetos de pruebas, que ha sido probada. Para que los resultados de las pruebas sean significativos y escrutables hasta el nivel de los ítems individuales, tenemos que saber la

versión de cada unidad que haya estado presente en un objeto particular de pruebas. Tenemos que ser capaces de relacionar cada ítem que hemos probado hacia atrás a los componentes conocidos del sistema, de versiones conocidas.

La entrega de una versión única y conocida del objeto de prueba. Nuevamente, para ser capaz de comprender el significado de los resultados de las pruebas, tenemos que ser capaces de entregar una versión de las pruebas en el laboratorio de pruebas.

Ahora, esta clase de comprensión a detalle no es algo que pasa por accidente. Tiene que ser planificado con anticipación. De este modo, durante la planificación del proyecto y de las pruebas, los procedimientos de la gestión de configuración e infraestructura (herramientas) deben ser seleccionados, documentados e implementados, de manera que no ocurran sorpresas en el momento de la ejecución de las pruebas.

¿Entonces, cuáles son las tareas clave de la gestión de configuración desde el punto de vista de las pruebas?

Tenemos que ser capaces de guardar y controlar el acceso a los ítems que componen el sistema. Este elemento de la gestión de configuración es llamado también control del código fuente, sin embargo, como puede suponer de la siguiente figura, esta tarea va más allá que sólo el código.

Porque tenemos que saber cuáles ítems de prueba estamos probando, el proceso de gestión de configuración debe ser capaz de identificar y documentar los ítems sometidos al control.

Porque los ítems estarán cambiando durante el proyecto, el proceso de gestión de configuración debe permitir el cambio de los ítems controlados a través de un proceso ordenado. Muchas organizaciones emplean comités de control de cambios para esta tarea, especialmente durante las últimas etapas de un proyecto porque llega a ser más difícil de anticiparse a las implicaciones de cualquier cambio dado por cualquier desarrollador.

El proceso de gestión de configuración debe permitir a alguien—a menudo denominado jefe de configuración o ingeniero de versiones—a informar acerca de los cambios pendientes, en camino y completados. El número de cambios debe llegar a un nivel muy bajo mientras nosotros llegamos a las etapas finales de las pruebas.

Finalmente, el proceso de la gestión de configuración nos debe permitir verificar que la implementación esté completa.

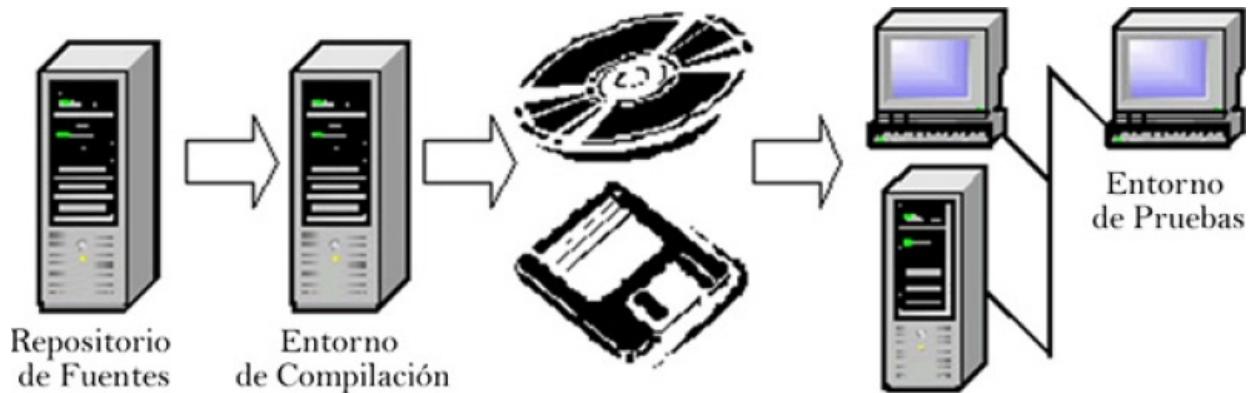


Figura 5.26: Gestión de Versiones de Pruebas

La gestión de versión de las pruebas es el proceso de transformar los ítems de pruebas que componen el sistema sometido a pruebas en un objeto de pruebas y luego transferir e instalar ese objeto de pruebas en el entorno de pruebas.

El proceso de gestión de versión de las pruebas debe especificar cuidadosamente cómo las siguientes tareas serán realizadas, quién qué papel toma y quién es responsable.

Primeramente, debemos tener algún tipo de proceso para determinar cuál objeto u objetos de pruebas deben ser liberados para las pruebas y con qué frecuencia. ¿Debe pasar esto semanalmente? ¿Diariamente? ¿Por horas? ¿Cada vez que le parezca a alguien?

Cuando una versión nueva del objeto de pruebas—a menudo denominada una construcción—tiene que ser instalada en el entorno de pruebas, ¿Cuál es el proceso para aplicar ese objeto actualizado de pruebas? ¿En otras palabras, cómo es instalada esa nueva construcción?

Algunas veces una nueva compilación crea problemas en el entorno de pruebas los cuales son suficientemente serios para justificar el retiro de ese objeto de pruebas. ¿Cómo retiramos una construcción deficiente e instalamos una que funcione o volvemos a una construcción anterior—pero que funcione?

Porque las pruebas deben ocurrir contra una versión conocida de la construcción o el objeto de prueba, ¿Cuál es el proceso utilizado para asignar un nombre único a cada construcción? Además, ¿De qué forma interrogan los probadores a la construcción para preguntar qué nivel de versión o de revisión es?

Finalmente, cuando se está probando familias complejas de sistemas o de sistemas de sistemas, necesitamos una manera de sincronizar con la base de datos y otros sistemas que interoperarán y cohabitarán con cada versión nueva. Ese proceso debe tomar en cuenta que estas bases de datos y sistemas también podrían estar cambiando.

Note que ésta es una compleja delegación de responsabilidades. La definición clara de los roles y las responsabilidades para cada paso es un **deber**.

Otra plantilla en el estándar IEEE 829 para la documentación es el informe de la transmisión de los ítems de pruebas. Es el único documento en ese estándar el cuál debe ser creado casi siempre por alguien diferente a un probador. La idea es que el equipo de ingeniería de versión creará este documento y lo enviará al probador junto con el objeto de prueba.

Un informe de la transmisión de los ítems de pruebas describe los ítems que deben ser entregados para las pruebas. La plantilla del estándar IEEE 829 para los informes de la transmisión de los ítems de pruebas incluye las secciones siguientes:

- El identificador del informe de la transmisión del ítem de pruebas.
- Los ítems transmitidos (incluyendo los nombres de los ítems y los números de revisión).
- El lugar del objeto o ítem de pruebas, incluyendo dónde está, de dónde vino, qué medios están siendo entregados, cómo ellos son etiquetados y alguna otra información necesaria para identificar de manera inequívoca el objeto o el ítem de pruebas para ser recogido y depositado en el entorno de pruebas.
- El estado del objeto o ítem de pruebas, incluyendo los defectos corregidos, los cambios introducidos, y así sucesivamente.
- Finalmente, si el objeto o ítem de pruebas está sometido al control de cambios, necesitamos saber quién aprobó ese objeto o ítem para la versión que debe ser probada.

Los informes de la transmisión de los ítems de pruebas son comúnmente llamados notas de versiones. A menudo, las notas de versiones sólo incluyen esta información y con frecuencia son documentos informales. El nivel de formalidad debe aumentar tanto como la complejidad—y así la probabilidad de los errores en la realización de una versión de pruebas—crece.

5.4.1 Ejercicios

Ejercicio 1

Identificar los principales ítems que usted piense que necesitarían gestión de configuración para Omninet.

¿Cuáles de estos ítems son importantes para la gestión de versión de pruebas?

Argumente.

Solución del Ejercicio 1

Para el centro de llamadas, el centro de datos, y el quiosco, la gestión cuidadosa de configuración del software como el hardware es esencial. Esto aplica a los servidores y las computadoras de escritorio en el centro de llamadas, así como también al hardware utilizado, personalizado y de distribución masiva.

Para ilustrar, por qué todo necesita gestión de configuración, considere esta anécdota. En un proyecto, ejecutamos pruebas de seguridad de un centro de datos durante las pruebas tempranas. Esa prueba reveló una lista de problemas, los cuales fueron entregados al desarrollo. Después, cuando el centro de datos había crecido, nosotros repetimos las pruebas. Cada pieza del equipo que estaba en el centro de datos, cuando la primera prueba fue ejecutada, pasó la prueba. Cada nueva pieza del equipo falló, más exactamente de la misma manera que el equipo original la tuvo la primera vez. La causa de los problemas, en todos los casos, no fue el software personalizado e instalado en los servidores, sino las opciones de configuración del sistema estándar las cuales no fueron correctamente definidas por los administradores del sistema.

5.5 Riesgo y Pruebas

Objetivos del Aprendizaje

LO-5.5.1 Describir un riesgo como un problema posible que amenazaría el alcance de uno o más objetivos del proyecto de los interesados del negocio. (K2)

LO-5.5.2 Recordar que los riesgos son determinados por medio de la probabilidad (de ocurrencia) y el impacto (el daño que resulta si es que ha ocurrido). (K1)

LO-5.5.3 Distinguir entre los riesgos de proyecto y producto. (K2)

LO-5.5.4 Reconocer los riesgos típicos de producto y proyecto.

LO-5.5.5 Describir y utilizar ejemplos, cómo el análisis y la gestión de riesgos pueden ser utilizados para la planificación de pruebas. (K2)

Esta sección, Riesgo y Pruebas, cubrirá los siguientes conceptos clave:

- Daños y riesgos potenciales.
- El riesgo como posible problema que amenaza el logro de los objetivos de los interesados del negocio.
- Riesgos de proyecto como opuestos a los riesgos de producto.

En el capítulo 4, introducimos la idea de las pruebas como una forma de gestión de los riesgos del producto o la calidad.

En la sección de este capítulo en la planificación de las pruebas, dijimos que el esfuerzo de las pruebas es un subproyecto del proyecto general. Por lo tanto, las pruebas están sujetas a riesgos, específicamente los riesgos de proyecto.

Porque el riesgo es la posibilidad de un resultado negativo, los riesgos de proyecto relacionados con las pruebas incluyen eventualidades como versiones de pruebas atrasadas, problemas del entorno de pruebas, y así sucesivamente.

Recuerde que la plantilla del plan de pruebas IEEE 829 nos pide identificar y evaluar no sólo los riesgos del producto, sino que también los riesgos del proyecto, específicamente aquellos riesgos del proyecto que pondrían en riesgo nuestra aptitud de realizar el subproyecto como planificado.

De este modo, para descubrir los riesgos del esfuerzo de las pruebas, pregúntese y pregunte a otros interesados del negocio:

¿Qué podría salir mal en el proyecto que retrasaría o invalidaría su plan y/o estimación de pruebas?

¿De qué tipo de resultados inaceptables de las pruebas se preocupa usted o se debe preocupar?

Por cada riesgo del proyecto—o de hecho cualquier riesgo—tiene cuatro opciones:

Una opción es la mitigación. Las acciones de mitigación son pasos preventivos que reducen la probabilidad o el impacto de un riesgo antes de que llegue a ser un evento o resultado real no deseado.

Otra opción es la contingencia. Las acciones de contingencia son pasos planificados con anticipación o reactivos que usted tomará para reducir el impacto de este evento o resultado, si el riesgo se convierte en un evento o resultado real no deseado. Note que las acciones de contingencia no pueden reducir la probabilidad, porque asumimos que el evento o resultado no deseado ha ocurrido ya.

Otra opción es la transferencia. La transferencia es básicamente una acción antes del evento en alguna otra parte para aceptar las consecuencias si el evento o el resultado no deseado en realidad ocurren. La clave de la transferencia es que la parte que acepta tiene que aceptar ambas consecuencias antes que el evento ocurra y si éste realmente ocurre.

Una opción final es de aceptar o ignorar el riesgo. Aquí no hacemos nada acerca del evento o resultado no deseado de antemano, lo cual es lo mejor si tanto la probabilidad como el impacto ¡son bajos!

Una forma de transferencia, la opción de gestión de los riesgos más típica en la vida diaria, es la compra de seguros. Sin embargo, en la mayoría de los proyectos de software o sistemas, esta opción no está disponible usualmente. Tome en cuenta que cuando se compra un seguro, se paga a algún tercero para aceptar alguna de las consecuencias, usualmente las financieras.

¿Cuáles son los riesgos de proyecto típicos para las pruebas? ¿Cuáles pasos de mitigación y/o contingencia podemos tomar para controlarlos?

Algunos riesgos se relacionan con los problemas de logística o calidad del producto que bloquean las pruebas. Aquí podemos usar una cuidadosa planificación, una buena clasificación de los defectos y un diseño de pruebas robusto.

- Otros riesgos relacionados con los entregables de pruebas que no se instalarán. Aquí podemos usar pruebas de humo de las compilaciones, compilaciones nocturnas y un proceso de desinstalación bien definido.
- Otros riesgos se relacionan con el cambio excesivo que invalida los resultados o necesita actualizaciones de las pruebas. Aquí podemos utilizar buenos procesos de control de cambios, robustos diseños de pruebas, corta documentación de las pruebas y escalamientos a la gerencia si las cosas se ponen demasiado difíciles.
- Otros riesgos se relacionan al entorno o los entornos de pruebas insuficientes o poco realistas. Aquí podemos explicarle a la gerencia, y, para los entornos complejos de las pruebas, algunas veces podemos tercerizar las pruebas sensibles al entorno como las pruebas de rendimiento.
- Otros riesgos se relacionan con el soporte poco fiable del entorno de pruebas. Aquí podemos definir buenos procesos de escalamiento del problema de tal forma que esos problemas no empeoren, o, mejor todavía, asegurar que tengamos las habilidades de la administración de los sistemas en el equipo de pruebas.
- Otros riesgos se relacionan con los vacíos en la cobertura de las pruebas que son reveladas durante la ejecución de las pruebas. Aquí podemos utilizar las pruebas exploratorias para detectar esos vacíos tempranamente y la mejora continua de las pruebas para reducir los

vacíos.

- Otros riesgos se relacionan con los descuidos en las fechas y/o los entregables que deben ser probados. Aquí podemos utilizar la prioridad del riesgo para eliminar las pruebas y la escalación a la gerencia para ayudarlos a entender los riesgos del producto creados por esas reducciones del alcance de las pruebas.
- Otros riesgos se relacionan con la reducción del presupuesto y/o el personal. Aquí podemos utilizar la prioridad del riesgo para eliminar las pruebas nuevamente, así como también la obtención de un equipo de pruebas bien formado o un aliado externo de pruebas para aumentar nuestras habilidades del equipo de pruebas.
- Otro riesgo se relaciona con la utilización del entorno de pruebas para la depuración. Aquí podemos escalar a la gerencia así que ellos puedan comprender el impacto de esas acciones, y utilizar la prioridad del riesgo para eliminar o rehacer el cronograma de las pruebas si es necesario.

Ampliamente hablando, ¿Cuáles son algunos otros tipos de riesgos de proyecto para las pruebas? Algunos riesgos surgen de factores organizacionales. Estos incluyen lo siguiente:

- Habilidades y reducción de personal.
- Cuestiones personales y de capacitación.
- Problemas de comunicación.
- Falta de seguimiento a los resultados de las pruebas, incluyendo las fallas para utilizar esos resultados para la mejora del proceso.
- Una actitud inapropiada hacia las pruebas o hacia las expectativas de las pruebas, si por medio de los probadores o por otros en el proyecto.
- Una organización compleja, tal como las organizaciones distribuidas o externalizadas.

Algunos riesgos surgen de cuestiones de los proveedores. Estos incluyen lo siguiente:

- Falla de un tercero, incluyendo un vendedor de herramientas de pruebas.
- Cuestiones contractuales.

Aún otros riesgos surgen de cuestiones técnicas. Estos incluyen lo siguiente:

- Problemas en la definición de los requisitos correctos.
- Requisitos inalcanzables (los cuales pueden conducir a pruebas bloqueadas o no ejecutables).
- Falta de calidad del diseño, el código, las pruebas y los datos de prueba.
- Un sistema altamente complejo.

5.5.1 Ejercicios

Ejercicio 1

Identificar de tres a cinco riesgos de proyecto para las pruebas de Omninet.

Evaluar la probabilidad y el impacto de cada riesgo en una escala de tres puntos (Alto, Medio o Bajo).

De los riesgos que ha identificado, elija uno o dos pasos apropiados que deben ser tomados para gestionar el riesgo.

Argumente.

Solución del Ejercicio 1

Riesgo alto: La logística o los problemas de calidad del producto bloquean las pruebas. El riesgo es alto por el gran número de componentes que deben ser integrados.

Mitigación: Pruebas rigurosas de componente e integración.

Contingencia: Diseñar “puntos de recuperación” en las pruebas donde las pruebas pueden ser iniciadas o continuadas, si fueron bloqueadas en algún otro lugar.

Contingencia: Tener una segunda fuente identificada para componentes críticos de distribución masiva.

Contingencia: Minimizar cualquiera de las dependencias de las pruebas de todas las componentes innecesarias; es decir, a menos que la prueba sea específicamente una prueba de todas las componentes, no se apoye en o utilice otros componentes para crear datos de prueba o para instalar condiciones de prueba.

Riesgo alto: El cambio excesivo invalida los resultados y requiere actualizaciones de pruebas. Las pruebas de usabilidad y beta resultarán en defectos que necesitan ser corregidos sin importar el impacto en las pruebas.

Mitigación: Colocar un buen proceso de control de cambios y una clasificación de defectos en vigencia de tal forma que los cambios sean juzgados basados en los riesgos, las oportunidades, los beneficios y los costos a través del proyecto entero.

Contingencia: Minimizar el alcance de la documentación de las pruebas para lo que es necesario para el equipo actual de tal manera que los cambios no necesiten demasiada edición.

Contingencia: Minimizar las dependencias entre las pruebas y todo los componentes para evitar mucho "daño colateral" cuando una prueba debe ser actualizada.

Riesgo medio: Los entregables de las pruebas no se instalarán. La mayoría de los componentes serán diseñados para la utilización en un entorno integrado, pero algunos serán personalizados.

Mitigación: Tener pruebas unitarias y compilaciones nocturnas (o más frecuente) automatizadas.

Contingencia: Definir procesos sólidos, fiables, de instalación y desinstalación, y pruebe aquellos procesos antes de empezar las pruebas.

Riesgo medio: El soporte del entorno de las pruebas es poco fiable. El entorno es complejo, que consiste de tres subsistemas principales muy diferentes.

Mitigación: Definir el apoyo de las prioridades para el equipo de pruebas durante la prueba de sistema.

Contingencia: Tener un administrador de sistemas de respaldo para el tratamiento de los trabajos de rutina en el equipo de pruebas.

Contingencia: Minimizar las dependencias entre las pruebas y todas las componentes para evitar un "daño colateral" cuando una prueba sea bloqueada debido a las cuestiones de disponibilidad del entorno.

5.6 Gestión de Defectos o Incidencias

Objetivos del Aprendizaje

LO-5.6.1 Reconocer el contenido de un informe de incidencia de acuerdo al 'Standard para la Documentación de las Pruebas de Software' (Estándar IEEE 829-1998). (K1)

LO-5.6.2 Escribir un informe de incidencia que cubre la observación de una falla durante las pruebas. (K3)

Glosario del ISTQB

Gestión de incidencias: El proceso de reconocer, investigar, tomando acción y disponiendo de las incidencias. Esto involucra las incidencias registradas, clasificándolas e identificando el impacto.

Esta sección, Gestión de Defectos o Incidencias, cubrirá los siguientes conceptos clave:

- El contenido de un informe de defecto o incidencia.
- Cómo escribir un informe de defectos o incidencia.

Un informe de defecto o incidencia es un formulario de comunicación escrita. Para cualquier forma de comunicación escrita, tenemos que considerar ambos las metas de esa comunicación y la audiencia de sus necesidades.

Los informes de incidencias o defectos tienen a menudo las siguientes metas:

- Proporcionar información detallada acerca de la incidencia o el defecto para quiénes lo necesitan.
- Ser parte de los datos agregados de los resultados de las pruebas para el análisis, el control y el informe.
- Conducir a la mejora del proceso de desarrollo y pruebas.

Los lectores típicos de los informes de defectos son:

- Desarrolladores: quien debe corregir el problema reportado.
- Jefes: quien debe realizar las asignaciones de los recursos y las decisiones de priorización acerca del problema.
- Personal de soporte técnico: quien debe estar consciente de los asuntos pospuestos o no resueltos en el momento del envío.
- Probadores: quien debe saber el estado actual del sistema.

Nosotros suponemos, cuando estamos creando un equipo de pruebas, que nuestros probadores no saben cómo comunicar por escrito.

Esa es la suposición más segura para un jefe de pruebas. Porque la comunicación escrita más visible, que un probador se dedica a escribir, es un informe de defecto, conviene que el jefe de pruebas enseñe esta habilidad. Convienen que el probador aprenda esta habilidad.

Aquí está el proceso de diez pasos que utilizamos para entrenar a los probadores a escribir buenos informes de defectos. Ha funcionado bien para nosotros.

1. Estructura: pruebe cuidadosamente.
2. Reproduzca: pruebe de nuevo.
3. Aísle: pruébelo de forma diferente.
4. Generalice: pruébelo en otro lugar.
5. Compare: revise los resultados de pruebas similares.

6. Resuma: relacione las pruebas con los clientes y usuarios.
7. Condense: elimine la información innecesaria.
8. Desambigüe: use palabras claras.
9. Neutralice: exprese el problema imparcialmente.
10. Revise: esté seguro de lo que está informando.

Revisemos estos pasos con más detalle. Sin embargo antes que entremos a detalles, déjenos apuntar que no estamos introduciendo un estilo o una guía de contenido aquí. Buenos informes de defectos acerca de un problema se pueden diferenciar en estilo y contenido, y eso está bien. Una de las grandes cosas acerca de escribir es que hay muchas maneras para contar la misma historia y llegar al mismo punto. La creatividad agobiante no debería ser una meta de los estándares para los informes de los defectos—o algunos otros productos del trabajo que son creados por las pruebas.

Primeramente, recuerde que las pruebas buenas, cuidadosas, orientadas al detalle y estructuradas son el fundamento para los buenos informes de defectos. Debería utilizar un método prediseñado para las pruebas.

Si está sentado frente a su teclado, conversando por escrito con un amigo, no prestando atención a lo que está haciendo y apretando sobre las teclas como un mono mareado, no resultarán buenos informes de defectos, porque usted no tendrá nada que transmitir al lector.

Entonces, si utiliza casos de prueba escritos, hágales el seguimiento o ejecútelo en forma automatizada de acuerdo con el proceso escrito o estandarizado para realizarlo de esa forma. Si está realizando pruebas exploratorias u otras basadas en la experiencia, utilice cartas de pruebas, una taxonomía de defectos, una lista de ataques o alguna otra guía para estructurar sus pruebas. Cuidadosamente tome notas acerca de lo que ve.

Cuando haga el seguimiento a los casos de prueba escritos, debe estar listo para expandir el caso de prueba cuando sea apropiado. A menudo le decimos a los probadores que el caso de prueba es un mapa de caminos a lugares interesantes, así que cuando llega a algún lugar interesante, pare y mire a su alrededor. Esta instrucción embebe la esencia de las pruebas exploratorias aún dentro del método de pruebas más automatizado.

Cuando ejecute una prueba, tenga los resultados esperados en mente. Si algunos resultados esperados son escritos en el caso de prueba, por supuesto, compruébelos. Sin embargo, algunas pruebas revelarán comportamientos que mientras no fueron específicamente puestos en la lista para que sean comprobados en el caso de prueba escrito, pueden ser problemáticos. Recuerde que el proceso de los informes de los defectos comienza cuando los resultados esperados son diferentes a los observados, y esté alerta por lo inesperado.

Sobre todo, la ejecución de una prueba es una tarea que necesita que su cerebro esté activo en todos los momentos. Las pruebas poco rigurosas resultan en informes de defectos poco rigurosos.

No todas las fallas son reproducibles, no todas las fallas no reproducibles no son importantes. De este modo, debemos comprobar siempre la reproducibilidad de la falla como parte de la escritura de un informe de defecto. Pensamos que tratando de reproducir la falla tres veces es una buena regla heurística.

La comprobación de la reproducibilidad le ayudará a documentar una secuencia bien definida de acciones que reproducirán la falla.

Porque las fallas no reproducibles pueden ser importantes, asegure de informar acerca de las fallas intermitentes, difíciles de repetir. Los problemas de rendimiento y de fiabilidad son notoriamente difíciles de reproducir, pero esas fallas pueden causar verdaderos problemas para los usuarios y clientes.

Habiendo intentado de reproducir la falla, ahora usted puede anotar la tasa de la incidencia de fallas en su informe. Por ejemplo, puede decirle a la gente que usted vio el problema solamente una vez en tres intentos. No solamente el cuerpo del informe de defecto debe mencionar el hecho de que la falla es intermitente, sino que también el resumen debe mencionar la intermitencia si existe.

Glosario del ISTQB

Tasa de fallas: La proporción del número de fallas de la categoría dada para un unidad de medida dada, p.ej. las fallas por unidad de tiempo, las fallas por el número de transacciones, las fallas por el número de ejecuciones en la computadora.

Un subproducto útil de esta comprobación para la reproducibilidad es que tendrá un conjunto limpio de pasos para reproducir la falla, si es posible. Esto aborda la cuestión de “no reproducibilidad” directamente. Porque muchos informes de defectos son rechazados como “no reproducibles”, identificando una falla discontinua como intermitente es esencial.

Algunas veces, la prueba misma es el problema. De este modo, debe aislar las anomalías observadas —discrepancias entre los resultados reales y los esperados—antes de la decisión de que un verdadero defecto existe.

Así, cuando reproduzca la falla, cambie las variables como las opciones o los datos de

configuración que podrían alterar el síntoma. Si es practicable, haga estos cambios uno por uno para asegurar que sabe cuál cambio es importante.

El aislamiento del proceso necesita razonamiento y comprensión del sistema sometido a pruebas. No podría ser inmediatamente obvio qué intentar. Teniendo su mente completamente comprometida es esencial aquí.

El aislamiento también puede ser extensivo, entonces sea cuidadoso de hacer concordar la cantidad del esfuerzo con la severidad y la prioridad de la falla observada. Pasando dos horas aislando un defecto trivial no es tiempo bien empleado, porque lo distrae a usted de tareas más importantes.

Adicionalmente, querrá evitar de meterse en actividades de depuración. Usualmente, como un probador, su trabajo es de enfocar en los comportamientos. Cuando comienza a examinar los detalles de la implementación—al menos que esté ejecutando pruebas de caja blanca o realizando pruebas de unidad como un desarrollador—entonces se ha metido probablemente en un área de trabajo que no es su responsabilidad.

Aislando un problema tiene también un subproducto útil. El buen aislamiento muestra la diligencia pendiente y les da a los desarrolladores un comienzo en la depuración. Eso puede disminuir la oportunidad de que su informe sea rechazado e incrementa la oportunidad de que el defecto sea corregido.

Algunas veces, la anomalía observada es simplemente un problema menor. Sin embargo, el defecto puede tener más manifestaciones serias. Así, debería buscar el caso general.

¿Hay fallas relacionadas en el sistema sometido a pruebas? ¿Ocurre la misma falla en otros módulos o lugares? ¿Hay más síntomas severos del mismo defecto?

Ahora, tiene que tener cuidado y evitar problemas confusos no relacionados. Eso puede conducir a informar un comportamiento como un defecto duplicado cuando es en realidad alguna cosa diferente. El mismo síntoma puede surgir de diferentes defectos.

Correctamente realizada, la generalización reduce el número de informes de defectos duplicados y refina su comprensión de la falla—y su habilidad de describir esa falla en su informe de defecto.

Si es posible, debería comparar los resultados que está viendo ahora contra los otros resultados de las pruebas.

Si ha ejecutado pruebas de ejecución similares, examine los resultados para la misma falla. ¿Ha ejecutado usted u otro probador la misma prueba contra las versiones anteriores? ¿Ha probado usted u otro probador condiciones similares, en otras pruebas, contra la misma versión?

Si esta prueba ha fallado ahora, donde funcionó antes, ¿Es la falla una regresión? Esa es una hipótesis razonable cuando un cambio introduce una falla no vista en versiones anteriores.

Es importante de mantener las restricciones de tiempo en mente aquí, como en el aislamiento. La comparación no es siempre posible. Por ejemplo, ¿Qué pasa si la prueba fue bloqueada previamente? ¿Qué pasa si la prueba no fue ejecutada por otras razones y una reinstalación de la versión anterior es impracticable? ¿Qué pasa si la característica probada no estaba disponible en versiones anteriores?

Intente comparar, pero evite de pasar más tiempo en esto si la severidad y la prioridad de la falla no garantiza el esfuerzo.

Habiendo investigado el problema, es ahora tiempo de pulir la descripción general. Parte de esto es poner un buen resumen acerca del problema.

Un resumen debería poner una “línea de etiqueta” en cada informe. Ésta debería capturar la falla y el impacto en el cliente, el usuario u otro interesado del negocio.

Una analogía útil para un resumen es el título del periódico. De hecho, una buena manera de aprender cómo escribir buenos resúmenes es leer el título en un artículo de un periódico y luego leer el artículo para ver si el autor ha capturado la esencia del artículo, lo cual es realmente importante, en el título.

Esto es más difícil de lo que parece. Sin embargo, los probadores deben invertir tiempo en escribir un buen resumen. Las ventajas de buenos resúmenes que incluyen la obtención de la atención de la gerencia y la ayuda a establecer las prioridades de las correcciones de los defectos. Adicionalmente, un buen resumen puede poner un nombre corto y descriptivo acerca de un informe de defecto para los desarrolladores. Recuerde que el resumen es a menudo la única parte del informe de pruebas que es leída en una reunión de la clasificación de los defectos o del comité del control de cambios.

Un buen informe de incidencia o defecto debería tener tan pocas palabras como sea posible, y no menos.

De este modo, debería leer nuevamente el informe y editarlo para eliminar las palabras o pasos extraños. Evite ambos, los comentarios en secreto y la conversación monótona acerca de puntos obvios. ¿Pregúntese a usted mismo, hay algunos detalles o acciones irrelevantes?

Recuerde, durante las semanas y días finales de un proyecto, el tiempo de cada uno espreciado.

No pierda ese tiempo obligando a que la gente lea palabras sin sustancia e innecesarias. Sin embargo, tampoco elimine algunos detalles esenciales.

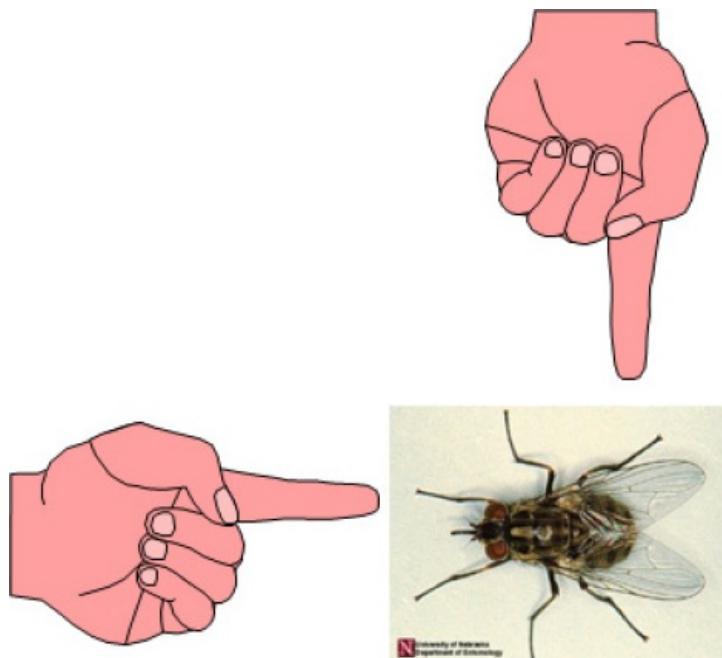


Figura 5.27: Desambiguar

Un buen informe de incidencia o de defecto utiliza exactamente las palabras correctas. Entonces, cuando esté afinando su informe de defecto, retire, reformule o expanda palabras o afirmaciones vagas, conducidas incorrectamente o subjetivas. Debe asegurar que el informe no esté sujeto a la incorrecta interpretación. Cada afirmación debería servir a la siguiente meta: Una afirmación clara e indisputable de hecho. Idealmente, su informe de la falla conducirá al desarrollador de la mano hacia el defecto.

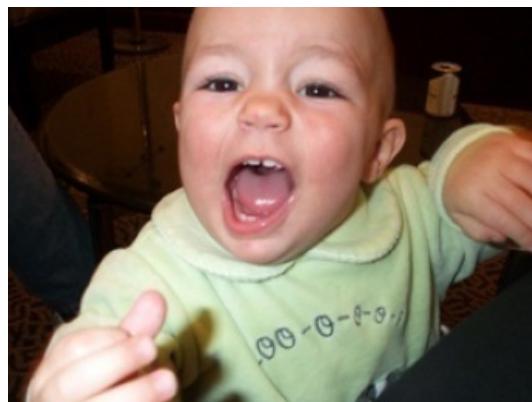


Figura 5.28: Neutralizar

Un chiste acerca de los probadores dice que el probador tiene el trabajo de decirle al programador que su bebe es feo. De este modo, un buen informe de incidencia o defecto debería ser neutral en el uso de las palabras.

Mientras que el informe de defecto es una importante manera en el cuál las pruebas producen oportunidades a la organización de ahorrar dinero, recuerde de entregar malas noticias gentilmente. Sea justo en el uso de sus palabras e implicaciones.

Hay un número de errores comunes de evitar aquí, incluyendo lo siguiente:

- Ataque a los desarrolladores o hacer declaraciones que podrían ser interpretadas como ataques.
- Criticando el error esencial y así mismo las habilidades del programador quién lo realizó.
- Mientras el humor es útil en muchos contextos, en la comunicación escrita, el humor o el uso del sarcasmo puede fácilmente parecerse a una agresión o una actitud negativa.
- Y, por supuesto, la expresión de frustración o el tener una rabiaacerca de la falla no será bien recibida.

Debería intentar de limitar los informes de defectos a afirmaciones de hecho.

Los informes de defectos, una vez escritos, son parte del registro permanente del proyecto. De este

modo, uno nunca sabe quién leerá los informes. Algunas veces, usted lamentará un comentario hecho en el calor del momento, especialmente si después ese comentario es extraído fuera del contexto.

Finalmente, hay una regla que utilizamos en nuestros equipos de pruebas: Dos pares de ojos. Antes que algún producto del trabajo sea considerado por terminado, alguien más debe revisarlo.

Entonces, cada probador debería presentar cada informe de defecto a uno o más colegas del equipo de pruebas para una revisión antes de que éste sea presentado. Esto no significa que se tiene que pasar horas trabajando en cada informe, sino más bien se tiene que pedir a un colega que compruebe el informe acerca de lo siguiente:

- ¿Hay formas de mejorar el informe?
- ¿Puede hacer el colega preguntas aclaradoras que conducirán a una mejor manera de expresarlas?
- ¿Podría ser el comportamiento correcto de verdad?

Recuerde que los informes de las pruebas son entregables clave del equipo de pruebas. En nuestras evaluaciones para clientes, a menudo encontramos que los informes deficientes de defectos son una mayor causa de fricción e inefficiencia. Entonces, el equipo de pruebas debería presentar solamente los mejores informes posibles de los defectos, dadas las restricciones del tiempo apropiadas para la prioridad y la severidad del defecto.

El estándar IEEE 829 para la documentación de las pruebas incluye una plantilla para los informes de incidencias.

Ahora, antes que hablemos acerca de la plantilla, esté seguro de mantener en mente la definición de una incidencia, una anomalía, una falla y un defecto. Una incidencia es una situación, donde durante las pruebas, es necesaria investigación adicional. Lo que causa una incidencia es a menudo una anomalía, una situación donde los resultados reales son diferentes de los esperados.

Algunas veces, una anomalía resulta de una falla. Una falla ha ocurrido cuando el ítem de prueba o el objeto de prueba se comportan incorrectamente debido a un defecto. Sin embargo, algunas veces una anomalía resulta de los problemas con el resultado esperado, los datos de prueba, el entorno de pruebas o las opiniones de los probadores acerca de lo que es un comportamiento correcto.

Un “bug”—el cuál es sinónimo de un defecto—es el problema esencial en el ítem de pruebas o el objeto de pruebas que causó la falla. En algunos casos, el defecto fue introducido en la codificación. Sin embargo en muchos otros casos el defecto fue introducido en las decisiones incorrectas del diseño o los requisitos.

Entonces, un informe de incidencia de las pruebas es un documento que describe un evento de pruebas que necesita investigación adicional. Esto es especialmente cierto si el comportamiento resulta de un defecto.

En el estándar IEEE 829, un informe de incidencia incluye las siguientes secciones:

El identificador del informe de incidencia de las pruebas.

Un resumen. Éste debería ser una línea, describiendo especialmente el impacto acerca de los interesados de negocios.

Una descripción de la incidencia. Esto debería cubrir las entradas, los resultados esperados, los resultados reales, la anomalía o las anomalías observadas en las entradas, la fecha y el horario cuando las anomalías fueron observadas, el entorno de pruebas en el cuál las anomalías fueron observadas, el identificador del caso de prueba o procedimiento que está en progreso cuando las anomalías fueron observadas, la reproducibilidad de las fallas, quién está informando la incidencia, y otros detalles como el impacto de la falla acerca de las pruebas, acerca del proyecto, acerca del producto, acerca de los interesados del negocio, y así sucesivamente.

Nuevamente, para enfatizar, estrictamente hablando, los informes de incidencia describen cualquier comportamiento cuestionable. Los informes de defectos describen el comportamiento debido a los defectos, los cuales son fallas. Un informe de defecto—a diferencia de un informe de incidencia—no debería describir pruebas o datos de prueba incorrectos, errores de los probadores, problemas del entorno de pruebas, y similares.

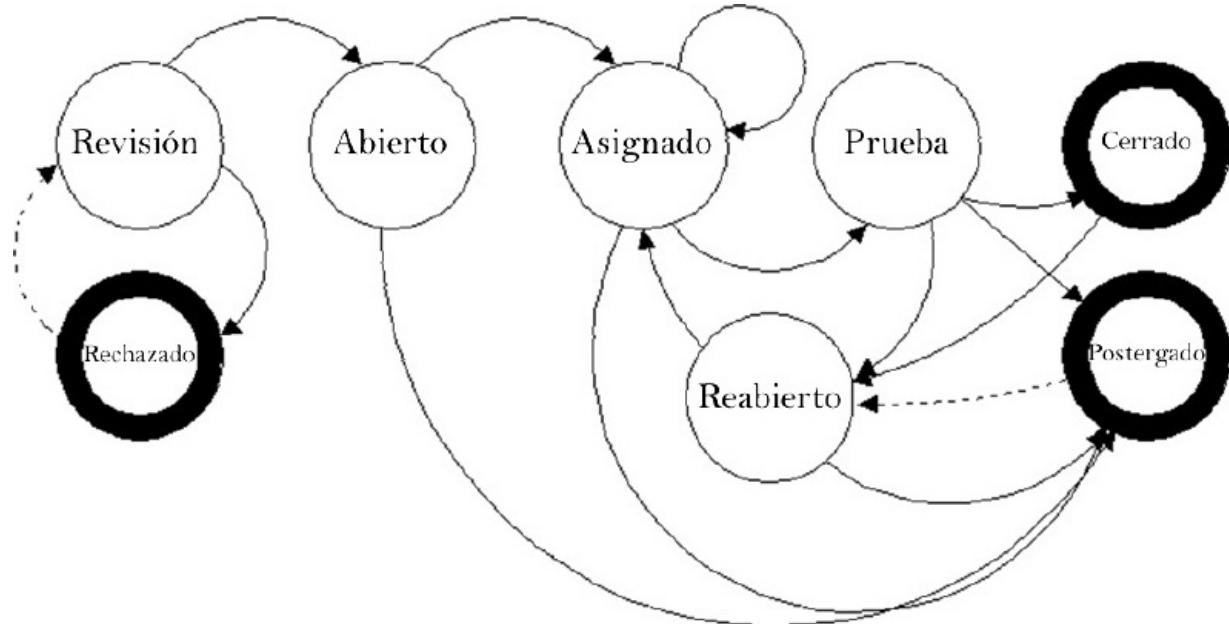


Figura 5.29: Ciclo de Vida o Flujo de Trabajo del Informe de Defecto

Una vez que el informe de incidencia es reportado, éste va generalmente a través de alguna secuencia de pasos hasta su resolución. Esta figura muestra un ejemplo de un posible diagrama de estados para la gestión de incidencias desde su descubrimiento hasta su resolución. Tome un momento para estudiar la figura.

En cualquier sistema de gestión de informes de defectos—el cuál debería ser construido en la herramienta de informes de defectos que usted utiliza—los informes de defectos se mueven a través de una serie de estados (en el ciclo de vida o flujo de trabajo) hasta su resolución.

En cada estado no terminal, un jefe o el comité de clasificación de defectos especifica un responsable quien es el que mueve el defecto al próximo estado. Los sistemas de seguimiento de defectos pueden y deberían implementar y automatizar estos flujos de trabajo, pero el equipo de trabajo y el soporte de la gestión realizan el trabajo del flujo de trabajo.

Bueno, antes que dejemos esta cuestión de los informes individuales de incidencias, ¿Qué otra información debería ser incluida en un informe de incidencia?

- La configuración del software o sistema.
- Si es posible, la fase de la introducción, la detección y la eliminación del defecto. Ésta es una buena idea, pero desafortunadamente realizada no frecuentemente. Los campos de la detección y de la eliminación son fácilmente llenados, pero el campo de la introducción es algunas veces difícil de determinar con exactitud, particularmente cuando surgen cuestiones de presión de plazos y culpabilidad.
- La urgencia o prioridad de corregir. Ahora, esto es obviamente un tema de opinión y las opiniones se diferenciarán, pero intentando de alcanzar un consenso en esto, ayuda a tomar decisiones inteligentes de lo que hay que corregir.
- Las conclusiones y recomendaciones recopiladas durante el ciclo de vida, tanto en cuanto a lo que hay que hacer inmediatamente con el defecto como los cambios a largo plazo que posiblemente queremos realizar para evitar la introducción de esos defectos en el futuro.
- Los riesgos, los costos, las oportunidades y los beneficios de la corrección o no corrección del defecto. Ésta es una buena manera de tomar decisiones inteligentes acerca de la postergación o la corrección de defectos individuales.
- La historia de los cambios, especialmente por cada cambio de estado, nos permite regresar y darnos cuenta de lo que le pasó al defecto.
- La fecha del informe, la empresa del informe y el autor, lo cual ayuda especialmente si tenemos preguntas acerca del informe.
- Los resultados esperados y reales, junto a una descripción clara de lo que es la diferencia entre estos dos si no es completamente obvio.
- La identificación clara del ítem y el entorno de pruebas. Con demasiada frecuencia, escuchamos esta respuesta, “Bueno, en mi sistema funcionó bien”, cuando reportamos los informes de defectos, y muchas veces es porque no fuimos lo suficientemente claros acerca de lo que estuvimos probando o acerca de los entornos en los que estuvimos probando.
- En cuál etapa del proyecto estuvimos y cuál proceso del ciclo de vida estuvo activo cuando la

incidencia fue observada.

- El alcance o grado del impacto en el(os) interesado(s) del negocio. Nuevamente, una consideración clave en cuanto a que si esto será pospuesto o en realidad corregido.
- La severidad del impacto en el sistema. Esto también debería tener otra consideración en cuanto a la postergación.

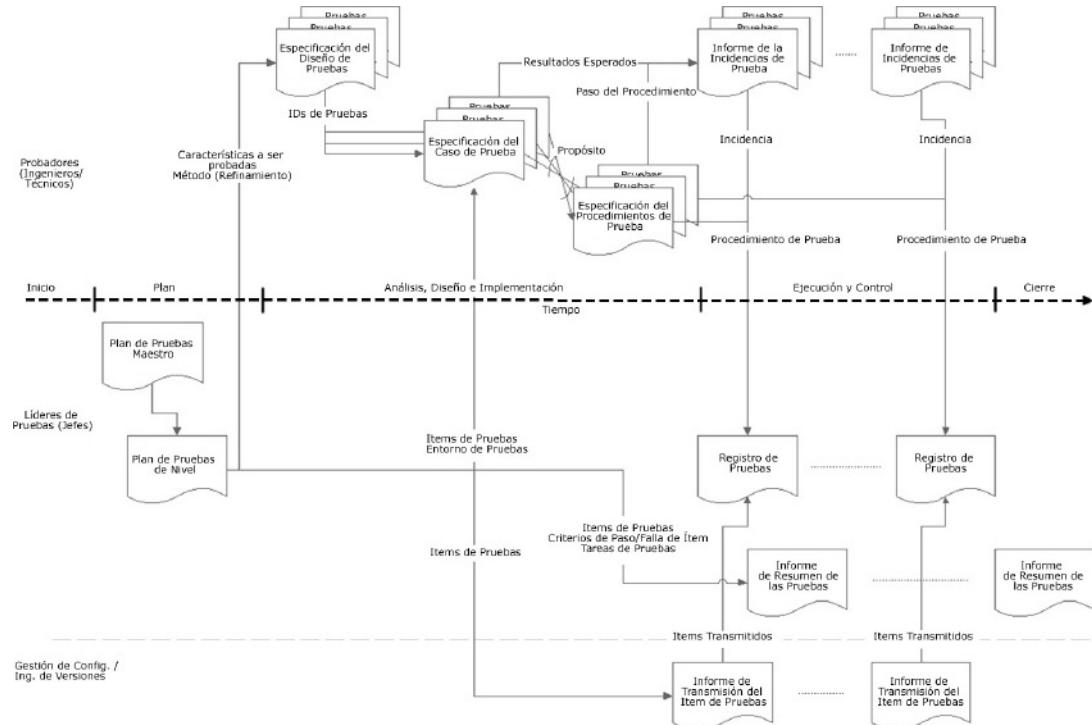


Figura 5.30: Plantillas de Documentación del IEEE 829

Ahora que hemos cubierto todas las plantillas de la documentación del estándar IEEE 829, examinemos una descripción general de esto. Pase breves momentos estudiando esta figura antes de que se lo describamos.

Note primero que, en el medio de la figura, mostramos la cronología del proyecto. Esto comienza con la iniciación del proyecto, luego la planificación, luego el análisis, el diseño y la implementación, luego la ejecución y el control de las pruebas y finalmente el cierre de las pruebas. Estos incluyen los pasos del proceso de pruebas básico descrito anteriormente. Esto pone cada documento del IEEE 829 en el contexto de un proyecto típico.

En el lado izquierdo de la figura, usted puede observar el rol asociado usualmente con la producción de un producto del trabajo mostrado en la derecha.

Durante el período de la planificación del proyecto, los líderes de pruebas producirán el plan maestro de pruebas y uno o más detalles o planes de pruebas de un nivel. En un método de pruebas basado en los riesgos, los planes se derivarán del análisis de los riesgos.

Ahora, los probadores analizarán los requisitos, el diseño y los riesgos. Ellos elaborarán acerca de las características que deben ser probadas e identificadas en los planes de pruebas. Una o más especificaciones del diseño de pruebas se originarán de este análisis.

Como los probadores diseñan los casos de prueba, ellos vincularán esos casos de prueba hacia atrás con las especificaciones del diseño de pruebas en cada especificación del caso de prueba. Usando identificadores de los casos de prueba, como se mostró anteriormente en este capítulo, harán posible ese vínculo. Las especificaciones del caso de prueba incluirán información acerca de los ítems de pruebas y los entornos de prueba que deben ser probados.

Las especificaciones del caso de prueba también incluirán los resultados esperados para los casos de prueba.

Basados en las especificaciones de los casos de prueba, el probador implementará las especificaciones de los procedimientos de prueba. Éstas incluyen los pasos de los procedimientos específicos. Las especificaciones de los procedimientos de prueba se vinculan con las especificaciones de los casos de prueba con el propósito del caso de prueba, él cual incluye los procedimientos que deben ser ejecutados contra ese caso de prueba. Para cada caso de prueba, uno o más procedimientos de prueba pueden implicar la utilización de ese caso, uno o más casos de prueba pueden ser asociados con un procedimiento de pruebas dado.

Cuando la ejecución de las pruebas comienza, recibiremos un objeto de pruebas, y, con la esperanza, un informe de transmisión del ítem de pruebas del ingeniero de versiones. Cada objeto

de prueba liberado al equipo de pruebas debería ser descrito por un informe de la transmisión de los ítems de pruebas.

Mientras que el probador ejecuta las pruebas, ellos registrarán los resultados de las pruebas en los registros de pruebas. Los registros de pruebas registran los procedimientos de prueba que están siendo ejecutados.

Durante la ejecución de las pruebas, los probadores informan usualmente las incidencias. Una serie de informes de incidencias de pruebas pueden ser llenados en un período de la ejecución de las pruebas. Los informes de incidencias de pruebas son vinculados con el objeto de prueba específico que está siendo probado. Los informes de incidencias de pruebas están vinculados hacia atrás con la especificación de los casos de prueba en cuanto a los resultados esperados. Los informes de incidencias de pruebas están vinculados hacia atrás con la especificación de los procedimientos de prueba en cuanto al procedimiento de prueba que está siendo ejecutado.

Durante la ejecución de las pruebas, el jefe de pruebas no solamente creará los registros de las pruebas, pero también producirá uno o más informes de resumen de las pruebas. La información en los registros de las pruebas hace referencia a los informes del resumen de las pruebas. Los informes del resumen de las pruebas también están vinculados hacia atrás con el plan de pruebas, basados en los ítems de pruebas, los criterios paso/falla de los ítems y las tareas de pruebas planificadas y reales.

Durante el cierre de las pruebas, el jefe de pruebas podría producir un informe final del resumen de las pruebas que resume los resultados de todo el período de la ejecución de las pruebas.

Examine esta figura hasta que el estándar le sea claro, porque este estándar de la documentación IEEE 829 es importante en los exámenes ISTQB Nivel Básico y Avanzado.

5.6.1 Ejercicios

Ejercicio 1

En la figura 5.31 se muestra un informe de defecto escrito incorrectamente del proyecto Omninet.

Trate de mejorarlo, haciendo algunas suposiciones razonables acerca de las pruebas realizadas.

Trate de cubrir todos los puntos abordados en el estándar del informe de incidencia IEEE 829.

Usted puede asumir que el instructor es el probador quién escribió el informe de defecto y haga preguntas para ayudar a llenar algún dato faltante.

Argumente.

Resumen: La sesión dura demasiado tiempo.

Pasos para reproducir:

1. Pasar la tarjeta de crédito en el subsistema de pago.
2. Comprar algunos períodos de tiempo.
3. Poner en marcha un cronómetro cuando la navegación comience.
4. La sesión durará entre uno a cinco minutos.

Figura 5.31: Informe de defecto escrito deficientemente

Solución del Ejercicio 1

El primer problema es que el resumen está incorrecto. ¿Qué significa exactamente la frase "demasiado tiempo"? La pregunta completa del impacto de negocios no es abordada y necesita ser abordada por la especificación acerca de cuánto tiempo se está dando. Si el sistema está dando 15 segundos en cada sesión, eso no es gran cosa. Si son 15 minutos, entonces sí. Este tema es cubierto después en el cuerpo del informe del defecto, pero muchos sistemas de seguimiento de defectos producen un informe de resumen, a menudo utilizado en las reuniones acerca de la clasificación de los defectos, en el cual sólo es mostrado el resumen de los mismos. Un resumen bien escrito describe el problema y su impacto.

El cuerpo del informe también plantea más preguntas que las que responde. ¿Cuántos períodos diferentes de tiempo fueron intentados? ¿Hay alguna relación entre el número de períodos comprados y el tiempo dado? ¿Cuántas veces intentó el probador este experimento? ¿Cuáles otros pasos para el aislamiento intentó el probador? Un buen informe de defecto deja al lector con respuestas, no con preguntas.

Glosario del ISTQB

Registro de incidencias: Guardado de los detalles de cualquier incidencia que ocurrió, p.ej. durante las pruebas.

Control de versión: Consulte el control de configuración.

Control de configuración: Un elemento de la gestión de configuración, que consiste en la evaluación, la coordinación, la aprobación o la desaprobación, e implementación de cambios a los ítems de configuración después de un establecimiento formal de su identificación de configuración. Note que este término no es específicamente enunciado en esta sección pero es incluido aquí porque en esta sección es un sinónimo del control de versión.

Riesgo de producto: Un riesgo relacionado directamente con el objeto de pruebas. Véase también riesgo. Note que en

este libro hemos utilizado el término riesgo de calidad como un sinónimo para el término riesgo de producto.

Riesgo de proyecto: Un riesgo relacionado con la gestión y el control del proyecto (de pruebas), p.ej. la falta de personal, los plazos estrictos, los requisitos que cambian, etc. Véase también riesgo.

Preguntas de Examen de Muestra y Simulación

Para finalizar cada capítulo, usted puede tratar de resolver una o más preguntas de examen de muestra para reforzar su conocimiento y comprensión del material y prepararse para el examen del Probador ISTQB Nivel Básico.

Sección 5.1 Organización de pruebas (K2)

Términos

Probador, líder de pruebas y jefe de pruebas.

#	Pregunta	K
138.	<p>Objetivo del aprendizaje: LO-5.1.1</p> <p>¿Cuál es una ventaja de emplear a probadores independientes?</p> <ul style="list-style-type: none">A. Los equipos de pruebas independientes son responsables de asegurar que se complete a tiempo el período de la ejecución de las pruebas.B. Los probadores independientes son a menudo más efectivos en encontrar defectos.C. No se necesita gente con conocimiento técnico o del dominio del negocio en un equipo de pruebas independiente.D. Las pruebas independientes liberan a los desarrolladores de cualquier responsabilidad de la calidad.	1
139.	<p>Objetivo del aprendizaje: LO-5.1.2 2</p> <p>A usted le gustaría formar un equipo de pruebas independiente para un próximo proyecto, con usted en el rol de líder. Sabe que los costos altos e impredecibles asociados con la necesidad de entregar los parches para las fallas de campo son una fuente principal de la preocupación para la compañía. ¿Cuál de los siguientes beneficios de las pruebas independientes podrían abordar efectivamente esta preocupación y promover su idea a la gerencia?</p> <ul style="list-style-type: none">A. Los probadores independientes ven otros y diferentes defectos que los probadores que no son independientes.B. Las pruebas independientes hacen el proceso de liberación más predecible.C. Los problemas de la calidad en el campo o área pueden ser peligrosos para la seguridad pública.D. Los probadores independientes tomarán la responsabilidad de los defectos.	2
140.	<p>Objetivo del aprendizaje: LO-5.1.3</p> <p>Usted ha sido nombrado líder de pruebas para todas las pruebas de un próximo proyecto. Actualmente está formando un equipo de pruebas para las pruebas de nivel de componentes. De las siguientes personas, ¿Quiénes estarían mejor calificadas para trabajar como probadores en este nivel?</p> <ul style="list-style-type: none">A. Los analistas de negocios.B. Los desarrolladores.C. Los usuarios expertos.D. Los operadores.	1
141.	<p>Objetivo del aprendizaje: LO-5.1.4</p> <p>¿Cuál de las siguientes es una tarea típica para un Líder de pruebas?</p> <ul style="list-style-type: none">A. Preparar y adquirir los datos de pruebas.B. Crear las especificaciones de las pruebas.C. Revisar y contribuir a los planes de pruebas.D. Coordinar la estrategia de pruebas y planificar con los jefes de proyecto y otros.	1
142.	<p>Objetivo del aprendizaje: término</p> <p>¿Qué es un jefe de pruebas?</p> <ul style="list-style-type: none">A. Una herramienta que facilita la grabación y el seguimiento del estado de los defectos.B. La persona, quien dirige, controla, administra, planifica y regula la evaluación de un objeto de prueba.C. Una herramienta que proporciona el soporte a la gestión de pruebas y a la	1

- parte del control de un proceso de pruebas.
- D. Un profesional calificado quien está involucrado en las pruebas de un componente o sistema.

Sección 5.2 Planificación y estimación de pruebas (K2)

Estándares

- [IEEE 829] Estándar IEEE 829™ (1998/2005) Estándar IEEE para la Documentación de las Pruebas de Software (actualmente en revisión)].

Términos

Método de pruebas.

#	Pregunta	K
143.	<p>Objetivo del aprendizaje: LO-5.2.1</p> <p>¿Qué cubre un plan maestro?</p> <p>A. Las pruebas en el nivel de pruebas de sistema. B. Las pruebas en el nivel de pruebas de aceptación. C. Todas las pruebas en el proyecto. D. Las pruebas de las versiones de mantenimiento.</p>	1
144.	<p>Objetivo del aprendizaje: LO-5.2.2</p> <p>Actualmente usted está trabajando en un proyecto como jefe de pruebas. Está pensando acerca de los criterios de entrada y salida y el entorno de pruebas, y argumentando acerca de las opciones con el equipo del proyecto. ¿Las decisiones tomadas acerca de estos temas a cuál documento pertenecen?</p> <p>A. Plan de pruebas. B. Especificación del diseño de pruebas. C. Especificación del procedimiento de prueba. D. Transmisión del ítem de prueba.</p>	2
145.	<p>Objetivo del aprendizaje: LO-5.2.3</p> <p>Temprano en un proyecto, cuando es publicado el primer borrador de la especificación de los requisitos, el jefe de pruebas de un proyecto le pide a usted que participe en una sesión de lluvia de ideas que identificará y evaluará los riesgos inherentes del producto. ¿Cuál método de pruebas es más probable que sea el método principal en ese proyecto?</p> <p>A. El contrario a la regresión. B. El consultivo. C. El analítico. D. El basado en modelos.</p>	2
146.	<p>Objetivo del aprendizaje: LO-5.2.4</p> <p>Considere las siguientes actividades:</p> <p>I. Determinar los riesgos que deben ser abordados por las pruebas. II. La instalación de un comité del control de cambios. III. Creación del cronograma de la ejecución de las pruebas. IV. Escribir un informe de incidencia. V. Seleccionar las métricas para el monitoreo y control de la ejecución de las pruebas.</p> <p>¿Cuál de las siguientes sentencias es verdadera?</p> <p>A. Todas las cinco actividades son actividades de la planificación de pruebas. B. Solo I, II, III, y V son actividades de la planificación de pruebas. C. Solo I, III, IV, y V son actividades de la planificación de pruebas. D. Solo I, III, y V son actividades de la planificación de pruebas.</p>	2
147.	<p>Objetivo del aprendizaje: LO-5.2.5</p> <p>Considere la siguiente lista de los casos de prueba, las prioridades (el número más pequeño es la más alta prioridad), y las dependencias:</p> <p>Número del caso de prueba: Nombre del caso de prueba/Prioridad/Dependencia/ 01.001: Navegar por los artículos /3/ninguno 01.002: Añadir un artículo al carro de compras /2/01.001 01.003: Pasar a pagar/1/01.002 01.004: Grabar el carro sin pasar a pagar /4/01.002</p>	3

	<p>01.005: Retornar a la tienda, acceder al carro guardado, pasar a pagar /4/01.004</p> <p>¿Cuál de los siguientes es un posible cronograma de la ejecución de las pruebas que considera ambos la prioridad y las dependencias?</p> <ol style="list-style-type: none"> 01.001, 01.002, 01.001, 01.002, 01.003, 01.004, 01.005 01.002, 01.001, 01.004, 01.005, 01.002, 01.001, 01.003 01.001, 01.002, 01.004, 01.005, 01.001, 01.002, 01.003 01.001, 01.002, 01.003, 01.001, 01.002, 01.004, 01.005 	
148.	<p>Objetivo del aprendizaje: LO-5.2.6</p> <p>¿Cuál de las siguientes es una actividad de de la ejecución de las pruebas que debería ser considerada durante la planificación de las pruebas?</p> <ol style="list-style-type: none"> Revisar la base de las pruebas. Integrar y coordinar las actividades de las pruebas en las actividades del ciclo de vida del software. Crear un cronograma de la ejecución de las pruebas. Finalizar y archivar el testware. 	1
149.	<p>Objetivo del aprendizaje: LO-5.2.7</p> <p>Considere los siguientes factores:</p> <ol style="list-style-type: none"> Las características del producto sometido a pruebas. El costo del reemplazo del entorno de pruebas existente. Las características del proceso de desarrollo. El resultado de las pruebas. Las barreras del lenguaje y las diferencias de zonas de horario entre los miembros del equipo. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ol style="list-style-type: none"> Los factores I, II y III influencian el esfuerzo de las pruebas en la mayoría de los proyectos. Todos estos factores influencian el esfuerzo de las pruebas en todos los proyectos. Los factores I, III y IV influencian el esfuerzo de las pruebas en la mayoría de los proyectos. Los factores III, IV y V influencian el esfuerzo de las pruebas en la mayoría de los proyectos. 	1
150.	<p>Objetivo del aprendizaje: LO-5.2.8</p> <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ol style="list-style-type: none"> estimación de las pruebas por los dueños de las tareas y los expertos es siempre más exacta que la estimación de las pruebas por medio de las métricas. estimación de las pruebas por los dueños de las tareas y los expertos puede ser utilizada junto con la estimación de las pruebas por medio de las métricas. estimación de las pruebas por los dueños de las tareas y los expertos es siempre menos exacta que la estimación de las pruebas por medio de las métricas. estimación de las pruebas por los dueños de las tareas y los expertos deberían preceder siempre a la estimación de las pruebas por medio de las métricas. 	2
151.	<p>Objetivo del aprendizaje: LO-5.2.4</p> <p>Usted está seleccionando métricas para el monitoreo del estado de los casos de prueba (pasados, fallados, aún no ejecutados, etc.), el estado de los defectos (encontrados, corregidos, corregidos confirmados, etc.), y el estado de la cobertura de pruebas (pruebas planificadas, pruebas ejecutadas y pasadas, pruebas ejecutadas y falladas, etc.) ¿Cuál tipo específico de planificación de las pruebas está usted realizando?</p> <ol style="list-style-type: none"> de pruebas para el proyecto. de pruebas para el nivel de pruebas de sistema. de pruebas para las pruebas de rendimiento. de pruebas para la ejecución de pruebas. 	2
152.	<p>Objetivo del aprendizaje: LO-5.2.6</p> <p>¿Cuál de las siguientes es una tarea específica para la preparación de las pruebas, que debería ser abordada durante la planificación de las pruebas?</p> <ol style="list-style-type: none"> el método global de las pruebas. y coordinar las actividades de pruebas en las actividades del ciclo de vida del software. decisiones acerca de cómo los resultados de pruebas serán evaluados y 	1

	<p>cuándo detener las pruebas.</p> <p>D. el monto, el nivel de detalle, la estructura y las plantillas para la documentación de las pruebas.</p>	
153.	<p>Objetivo del aprendizaje: LO-5.2.9</p> <p>Suponga que encuentra el siguiente criterio de entrada en el plan de pruebas de aceptación:</p> <p>El sistema debería estar completo con las características, y con todos los defectos que deben ser corregidos y confirmados como corregidos por el equipo de pruebas del sistema, antes de la entrega de los primeros ítems para la prueba de aceptación.</p> <p>¿Cuál de las siguientes razones podría justificar este criterio de entrada?</p> <p>A. probable que el sistema sea realizado incorrectamente sin ese criterio.</p> <p>B. prueba de aceptación será realizada por usuarios expertos, cuyo tiempo es muy valioso y quienes no deberían ser frustrados haciendoles probar sistemas defectuosos e incompletos.</p> <p>C. meta de la prueba de aceptación es encontrar tantos defectos como sea posible.</p> <p>D. meta de la prueba de aceptación es prevenir tantos defectos como sea posible.</p>	2
154.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es un método de pruebas?</p> <p>A. fuente para determinar los resultados esperados para comparar con el resultado real del software sometido a pruebas.</p> <p>B. documento que especifica una secuencia de acciones para la ejecución de una prueba.</p> <p>C. implementación de la estrategia de pruebas para un proyecto específico.</p> <p>D. documento que especifica las condiciones de las pruebas (la cobertura de ítems) para un ítem de pruebas, el método de pruebas detallado y la identificación de los casos de prueba de alto nivel asociados.</p>	1
155.	<p>Objetivo del aprendizaje: Estándar IEEE 829</p> <p>¿Cuál de las siguientes es una sección principal en la plantilla del plan de pruebas del IEEE 829?</p> <p>A. Asignación de personal y necesidades de capacitación.</p> <p>B. Pasos de procedimiento.</p> <p>C. Especificaciones de las entradas.</p> <p>D. Criterios paso/falla.</p>	1

Sección 5.3 Monitoreo y control del progreso de pruebas (K2)

Estándares

- [IEEE 829] Estándar IEEE 829™ (1998/2005) Estándar IEEE para la Documentación de las Pruebas de Software (actualmente en revisión)].

Términos

Densidad de defectos, tasa de fallas, control de pruebas, monitoreo de pruebas e informe del resumen de las pruebas.

#	Pregunta	K
156.	<p>Objetivo del aprendizaje: LO-5.3.1</p> <p>¿Cuál de las siguientes son métricas comunes utilizadas para monitorear la ejecución de pruebas?</p> <p>A. El porcentaje preparado de los casos de prueba planificados.</p> <p>B. El porcentaje completo de la configuración del entorno de pruebas.</p> <p>C. El porcentaje de la cobertura de las bases de pruebas por las pruebas preparadas.</p> <p>D. El número acumulativo de los defectos, ambos encontrados y resueltos.</p>	1
157.	<p>Objetivo del aprendizaje: LO-5.3.2</p> <p>Usted está trabajando como un jefe de pruebas a cargo de las pruebas de sistema e integración en un proyecto, que está a medio camino a lo largo de un período planificado en el cronograma de la ejecución de pruebas. Considere el siguiente gráfico, que muestra los defectos acumulados, abiertos y cerrados en el proyecto durante el tiempo:</p>	2

Ejecución de las Pruebas de Integración y Sistema
Análisis de los Problemas de Calidad del Sistema



Suponga que el jefe de marketing del producto sugiere, que el producto actualmente bajo pruebas debería ser publicado inmediatamente a los clientes existentes para evitar hacerlos esperar tres meses más con tal producto defectuoso, porque la versión actual sufre de 500 problemas conocidos en el área o campo. Utilizando **sólo** el gráfico mostrado arriba, ¿Cuáles datos puede Ud. mostrar al equipo del proyecto, que podrían cambiar sus ideas acerca de esta sugerencia?

- A. Demasiadas pruebas están todavía fallando para poder liberar ahora, y un gran número de pruebas se relacionan con defectos críticos.
- B. Las pruebas ejecutadas no cubren de manera suficiente la base de las pruebas, y un gran número de incógnitas riesgosas quedan para ser abordadas.
- C. El número de defectos pendientes es mayor que los 500 defectos en el área o campo.
- D. Las pruebas han estado revelando cerca de 150 defectos nuevos por semana en las últimas cinco semanas, y Ud. espera encontrar alrededor de 750 defectos más antes del final del proyecto.

158.	<p>Objetivo del aprendizaje: LO-5.3.3</p> <p>Considere la siguiente sección de resumen de un documento IEEE829 del informe del resumen de las pruebas entregado después que las pruebas fueron finalizadas para un producto de entretenimiento de aparatos electrónicos de consumo:</p> <p>Hemos ejecutado el 100% de las 547 pruebas planificadas, y hemos complementado esas pruebas con 83 pruebas exploratorias adicionales para buscar otros defectos, que nuestros guiones no podrían encontrar de otra manera. Ninguna de las pruebas de guión o exploratorias tienen algún defecto pendiente, conocido, o defectos que deben ser corregidos y registrados contra estas. Nosotros trazamos los guiones de pruebas hacia atrás a la especificación de los requisitos, la especificación del diseño y el análisis de riesgos de calidad, y no hay omisiones conocidas de la cobertura de pruebas. Hemos encontrado 235 defectos, de los cuales todos han sido resueltos. 127 fueron considerados como imprescindibles de corregir y reparar; hemos confirmado todas esas reparaciones por medio de la repetición de pruebas. Los otros defectos fueron postergados (92) por el equipo de gestión del proyecto por no ser suficientemente importantes para demorar la versión o fueron cancelados (16) por que no son desviaciones del comportamiento deseado.</p> <p>Basado en esta información, ¿qué esperaría usted que el jefe de pruebas concluya?</p> <ul style="list-style-type: none"> A. La cobertura de pruebas fue insuficiente para apoyar una decisión de liberación. B. Demasiados defectos conocidos existen para apoyar una decisión de liberación. C. Las pruebas exploratorias fueron utilizadas por el propósito equivocado. D. Todos los datos disponibles apoyan una decisión de liberación. 	2
159.	<p>Objetivo del aprendizaje: término <i>¿Qué significa tasa de fallas?</i></p>	1

	<p>A. La desviación del componente o sistema de su entrega esperada, servicio o resultado esperado.</p> <p>B. La proporción del número de fallas de una categoría dada para una unidad dada de medida.</p> <p>C. El número de defectos identificados en un componente o sistema dividido por el tamaño del componente o sistema.</p> <p>D. Reglas de decisión utilizadas para determinar si un ítem de pruebas (función) o característica ha pasado o fallado una prueba.</p>	
160.	<p>Objetivo del aprendizaje: Estándar IEEE 829</p> <p>¿Cuál de las siguientes es un título de sección de la plantilla del registro de las pruebas IEEE?</p> <p>A. Registro de actividades y eventos.</p> <p>B. Criterios de paso/falla de los ítems.</p> <p>C. Pasos de procedimiento.</p> <p>D. Especificación de salidas.</p>	1

Sección 5.4 Gestión de Configuraciones (K2)

Estándares

- [IEEE 829] Estándar IEEE 829™ (1998/2005) Estándar IEEE para la Documentación de las Pruebas de Software (actualmente en revisión).

Términos

Gestión de configuración y control de versiones.

#	Pregunta	K
161.	<p>Objetivo del aprendizaje: LO-5.4.1</p> <p>Usted está probando un software de videojuego que se ejecuta en un hardware estándar, disponible comercialmente. El equipo de desarrollo informa, que no puede reproducir las fallas que usted está observando en el laboratorio de pruebas. Su proyecto no cuenta con una herramienta de gestión de configuración o de procesos. ¿Cuál de los siguientes es un beneficio de introducir esas herramientas y esos procesos, que podría ayudar a reducir la incidencia de este problema?</p> <p>A. Asegurar que el hardware de las pruebas y el hardware del desarrollo son los mismos.</p> <p>B. Determinar si las fallas son intermitentes cuando son informadas.</p> <p>C. Identificar los ítems de pruebas como únicos.</p> <p>D. Proporcionar la trazabilidad de las pruebas a los requisitos.</p>	2
162.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es el control de versiones?</p> <p>A. Un grupo de gente responsable de la evaluación y la aprobación o desaprobación de los cambios propuestos para los ítems de configuración, y para asegurar la implementación de los cambios aprobados.</p> <p>B. Un elemento de la gestión de configuración, que consiste de la evaluación, coordinación, aprobación y desaprobación, e implementación de los cambios en los ítems de configuración.</p> <p>C. Una agregación de hardware, software o de ambos, que es designada para la gestión de configuración y tratada como una sola entidad en el proceso de gestión de configuración.</p> <p>D. La composición de un componente o sistema definido por el número, la naturaleza y las interconexiones de sus partes constituyentes.</p>	1
163.	<p>Objetivo del aprendizaje: Estándar IEEE 829.</p> <p>¿Cuál de las siguientes es una sección importante en el informe IEEE829 de la transmisión de los ítems de pruebas?</p> <p>A. Criterios de entrada.</p> <p>B. Pasos de procedimiento.</p> <p>C. Condiciones de pruebas.</p> <p>D. Ubicación.</p>	1

Sección 5.5 Riesgos y Pruebas (K2)

Términos

Riesgo de producto, riesgo de proyecto, riesgo y pruebas basadas en el riesgo.

#	Pregunta	K
164.	<p>Objetivo del aprendizaje: LO-5.5.1</p> <p>Usted está trabajando como un jefe de pruebas en la primera versión de un nuevo paquete de software de automatización de oficinas para computadores personales. Usted ha escrito exitosamente un plan de prueba, ha preparado sus pruebas, ha formado su equipo de pruebas, y ha configurado su entorno de pruebas. ¿Cuál de los siguientes es el riesgo de proyecto más probable que impacte su capacidad de llevar a cabo su plan de pruebas?</p> <ul style="list-style-type: none"> A. Los defectos que afectan a los clientes existentes. B. Los defectos que afectan al rendimiento del producto. C. Los defectos que afectan la instalabilidad del producto en el entorno de pruebas. D. La indisponibilidad de los probadores calificados apropiados. 	2
165.	<p>Objetivo del aprendizaje: LO-5.5.2</p> <p>Consideré los siguientes factores relacionados con los riesgos del producto:</p> <ol style="list-style-type: none"> I. La probabilidad de ocurrencia. II. El número de riesgos del producto. III. El impacto de la falla. IV. La trazabilidad de los riesgos a los requisitos. V. El número de personas involucradas en el análisis de riesgos. <p>Para cualquier ítem del riesgo del producto individual el cual ha sido identificado, ¿Cuál de los siguientes conjuntos de factores afectan directamente al nivel de riesgo?</p> <ul style="list-style-type: none"> A. I y III B. I, II y III C. Todos los cinco factores. D. III, IV y V 	1
166.	<p>Objetivo del aprendizaje: LO-5.5.3</p> <p>Usted está trabajando como jefe de pruebas en un proyecto de banca en línea. ¿Cuál de los siguientes es un riesgo del producto para su proyecto?</p> <ul style="list-style-type: none"> A. Su proveedor de desarrollo tercerizado falla en la entrega a tiempo del software. B. No puede obtener suficiente experticia del dominio en su equipo de pruebas. C. Los hallazgos del equipo de pruebas son ignorados por el equipo del proyecto. D. El software no hace correctamente un seguimiento de los retiros de las cuentas bancarias de los clientes. 	2
167.	<p>Objetivo del aprendizaje: LO-5.5.4</p> <p>¿Cuál de los siguientes es un riesgo de proyecto típico?</p> <ul style="list-style-type: none"> A. La falla de un proveedor en la entrega a tiempo del hardware de pruebas. B. Los problemas de rendimiento en el software sometido a pruebas. C. Los problemas de usabilidad en el software sometido a pruebas D. La criticidad relacionada con la seguridad de los problemas potenciales. 	1
168.	<p>Objetivo del aprendizaje: LO-5.5.5</p> <p>Usted está trabajando como un jefe de pruebas en un proyecto nuevo e interno de software bancario. Durante el período de la planificación de las pruebas, usted reúne a los interesados clave del negocio para identificar y priorizar los riesgos clave del producto para el software. Durante el período de preparación de las pruebas, ¿qué le permitirá hacer a usted este análisis de riesgos del producto?</p> <ul style="list-style-type: none"> A. Determinar el grado de las pruebas para cada riesgo. B. Anticipar las cuestiones de la comunicación de los probadores. C. Informar los resultados de la ejecución de pruebas en cuanto a los riesgos, los cuales han sido cubiertos por las pruebas. D. Preparar un informe del resumen de las pruebas que describe el riesgo, el cual no fue abordado durante las pruebas. 	2
169.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es un riesgo del producto?</p> <ul style="list-style-type: none"> A. Un riesgo relacionado a la gestión y control del proyecto. B. Un riesgo directamente relacionado al objeto de prueba. 	1

- C. Un factor que podría resultar en consecuencias futuras negativas.
 D. La probabilidad y el impacto.

Sección 5.6 Gestión de incidencias (K3)

Estándares

- [IEEE 829] Estándar IEEE 829™ (1998/2005) Estándar IEEE para la Documentación de las Pruebas de Software (actualmente en revisión)].

Términos

Registro de incidencias y gestión de incidencias.

#	Pregunta	K
170.	<p>Objetivo del aprendizaje: LO-5.6.1</p> <p>¿Cuál de los siguientes pertenecen a la sección de la descripción de la incidencia de un informe de incidencia conforme al estándar IEEE 829?</p> <p>A. Los criterios paso/falla de los ítems de pruebas. B. Los resultados esperados y los resultados reales de una prueba. C. Todos los riesgos del proyecto. D. El grado en el cual un sistema o componente logra sus funciones designadas con las restricciones dadas en relación con el tiempo de procesamiento y la tasa de tráfico.</p>	1
171.	<p>Objetivo del aprendizaje: LO-5.6.2</p> <p>El siguiente resumen se encuentra en un informe de defecto presentado para un sistema de comercio electrónico:</p> <p>El sistema no acepta los números de las tarjetas Visa o MaterCard como válidos.</p> <p>¿A cuál de los siguientes objetivos o entregables de información de un informe de incidencia satisface este extracto más directamente?</p> <p>A. Proporcionar a los desarrolladores una visión acerca de que si el problema es reproducible. B. Proporcionar a los desarrolladores una visión acerca de cuántas pruebas son bloqueadas por la falla. C. Proporcionar a la gerencia una visión acerca de la importancia de la reparación del problema. D. Proporcionar al equipo del proyecto una visión acerca de que si el problema está actualmente resuelto.</p>	3
172.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es una incidencia?</p> <p>A. Un desperfecto en un componente o sistema que puede hacer que el componente o sistema falle para realizar su función requerida. B. Cualquier evento que ocurra y que necesita investigación. C. La desviación del componente o sistema de su entrega, servicio o resultado esperado. D. La manifestación física o funcional de una falla.</p>	1
173.	<p>Objetivo del aprendizaje: Estándar IEEE 829.</p> <p>¿Cuál de las siguientes es una sección importante en la plantilla de informes de incidencia del IEEE 829?</p> <p>A. Criterios de pruebas. B. Evaluación comprensiva. C. Ubicación. D. Impacto.</p>	1

Capítulo 5 Pregunta de todas las secciones

#	Pregunta	K
174.	<p>Cubre: las secciones 5 y 6</p> <p>Usted está trabajando como un jefe de pruebas en un proyecto de banca en línea. Se le entregó una extensiva especificación de los requisitos que aborda todas las características de calidad importantes. Ha ejecutado las pruebas que cubren cada</p>	2

riesgo significativo identificado durante el proceso del análisis de los riesgos del producto. También ha empleado a los probadores experimentados y usuarios expertos para ejecutar las pruebas exploratorias a través de todas las características de funcionalidad, rendimiento, seguridad y usabilidad del producto. Usted genera un informe que demuestra que las pruebas exploratorias de rendimiento identifican muchas más fallas que lo que fue anticipado durante el análisis de los riesgos del producto.

¿Cuál es la probable conclusión que se puede sacar de este informe?

- A. Las fallas del rendimiento no son de alto impacto.
- B. El análisis de los riesgos del producto subestimó la probabilidad de las fallas del rendimiento.
- C. Los probadores exploratorios se confundieron acerca de la importancia relativa de la seguridad comparada con el rendimiento.
- D. Hay pocos riesgos que quedan para el sistema relacionados con la usabilidad.

Preguntas del Examen de Simulación 1

#	Pregunta	K
175.	<p>Objetivo del aprendizaje: LO-5.6.2</p> <p>Considere el siguiente extracto de un informe de defecto:</p> <p>Para recrear la falla, utilizamos el archivo de pruebas TS01TC072.dat, el cual está disponible en el directorio compartido de las pruebas.</p> <p>¿Cuál objetivo del informe de incidencia satisface este extracto?</p> <ul style="list-style-type: none"> A. Proporcionar información a los líderes de las pruebas para informar acerca del progreso de las pruebas. B. Proporcionar información a los desarrolladores para aislar la falla. C. No pertenece a un informe de incidencia. D. Proporcionar ideas para la mejora del proceso de pruebas. 	3
176.	<p>Objetivo del aprendizaje: LO-5.1.1</p> <p>Usted está trabajando como el jefe de un equipo de pruebas independiente. En una reunión del proyecto, usted está explicando los resultados de sus pruebas hasta ahora. Usted muestra al equipo, que mientras las pruebas avanzan productivamente, está tomando un tiempo en resolver algunos defectos y fallas de las pruebas. Otro jefe comenta que le preocupa que el equipo de pruebas esté demorando la versión del software. ¿Cuál de las siguientes es una desventaja de las pruebas independientes ilustrada por este comentario del jefe?</p> <ul style="list-style-type: none"> A. El equipo de pruebas es considerado como el responsable por los retrasos. B. El equipo de pruebas está aislado del resto del equipo del proyecto. C. Los probadores independientes ven otros y diferentes defectos, y son imparciales. D. Los probadores independientes pueden verificar las suposiciones que otras personas hicieron. 	2
177.	<p>Objetivo del aprendizaje: LO-5.2.7</p> <p>Considere la siguiente lista de estrategias típicas de las pruebas:</p> <ol style="list-style-type: none"> I. Analítica. II. Basada en el modelo. III. Metódica. IV. Compatible con un estándar V. Heurística. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Todas las cinco estrategias, I, II, III, IV y V, son inherentemente estrategias reactivas. B. Todas las cinco estrategias, I, II, III, IV y V, son inherentemente estrategias preventivas. C. III, IV, y V son inherentemente reactivas. D. Sólo la V es inherentemente reactiva. 	1
178.	<p>Objetivo del aprendizaje: LO-5.3.2</p> <p>Usted está trabajando como un jefe de pruebas. El proyecto implica el desarrollo de un software para un cliente, que también es el usuario final. Usted está a cargo de la prueba de sistema. Uno de los juegos de pruebas que su equipo ha ejecutado durante la prueba de sistema, es el juego de pruebas de aceptación, el cual está basado en los requisitos del cliente que están contractualmente definidos. Los usuarios con más experiencia del cliente ejecutarán el conjunto de</p>	2

	<p>pruebas de aceptación en el momento de la entrega. El pago final y la aceptación del software por el cliente están sujetos a la exitosa finalización de estas pruebas. En una reunión del proyecto cerca del final del proyecto, usted informa que 15 de las pruebas de aceptación, alrededor del cinco por ciento del número total de las pruebas de aceptación, fallaron. Si le preguntan a usted en la reunión por qué el equipo debería estar preocupado acerca de estas pruebas falladas, ¿cuál de las siguientes podría ser una respuesta razonable?</p> <ul style="list-style-type: none"> A. "No importa, realmente, porque los clientes estarán satisfechos mientras pasen el 80% de las pruebas". B. "Todo software debería ser entregado siempre completamente libre de defectos, y este software no cumple ese estándar". C. "Dependiendo de la reacción del cliente acerca de algunos de los defectos que afectan esas pruebas, podríamos tener problemas contractuales con la aceptación y el pago final". D. "Oh, lo siento, no debería haber mencionado esas pruebas y sus estados, porque no es un dato apropiado del seguimiento del progreso de las pruebas". 	
179.	<p>Objetivo del aprendizaje: LO-5.4.1</p> <p>¿Cuál de las siguientes es una manera en la cual la gestión de configuración apoya a las pruebas?</p> <ul style="list-style-type: none"> A. Identificar el ítem probado como único. B. Asegurar que el código haya sido probado con pruebas de unidad. C. Determinar automáticamente la ubicación de algunos defectos encontrados. D. No hay beneficios; la gestión de configuración está fuera del alcance de las pruebas. 	1
180.	<p>Objetivo del aprendizaje: LO-5.5.3</p> <p>Usted está trabajando en un proyecto de comercio electrónico. Durante la parte del análisis del proceso de pruebas, usted se preocupó acerca de una posible respuesta lenta a los pedidos del cliente cuando el sistema está bajo carga normal. ¿Cuál de las afirmaciones siguientes describe mejor la situación?</p> <ul style="list-style-type: none"> A. El rendimiento de la aplicación no cumplirá los requisitos durante las pruebas. B. Usted ha identificado un riesgo del producto que debería ser considerado para las pruebas. C. Usted ha identificado un riesgo del proyecto el cual es improbable que afecte a las pruebas. D. El tiempo de respuesta no es un atributo de calidad para las aplicaciones de comercio electrónico. 	2
181.	<p>Objetivo del aprendizaje: LO-5.6.1</p> <p>Considere los siguientes fragmentos de información:</p> <ol style="list-style-type: none"> I. Las entradas suministradas. II. Los resultados reales observados. III. Los resultados esperados con estas condiciones. IV. El entorno utilizado. V. La reproducibilidad del comportamiento. <p>¿Cuál de las siguientes afirmaciones es la más precisa si estamos hablando acerca de un informe de incidencia que cumple con el estándar IEEE 829?</p> <ul style="list-style-type: none"> A. Ninguna de esta información pertenece a un informe de incidencia. B. Los ítems I, II y III pertenecen a un informe de incidencia. C. Toda esta información pertenece a un informe de incidencia. D. Los ítems I, II, IV y V pertenecen a un informe de incidencia. 	1
182.	<p>Objetivo del aprendizaje: LO-5.6.2</p> <p>Considere el siguiente extracto de un informe de defecto:</p> <p>Ésta es la tercera vez que este defecto ha vuelto a aparecer. Cada vez que nos topamos con este problema en las pruebas, nos cuesta de 3 a 5 horas hombre de esfuerzo. Realmente esto es ridículo. Ustedes, los desarrolladores están matando nuestra productividad.</p> <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Este extracto pertenece a la sección del impacto de un informe de incidencia del IEEE 829. B. Este extracto pertenece a la sección de la descripción de la incidencia de un informe de incidencia del IEEE 829. C. Este extracto pertenece al registro de las pruebas del IEEE 829 y no a un informe de incidencia del IEEE 829. D. Este extracto no cumple con las mejores prácticas en cuanto a la comunicación entre el probador/desarrollador. 	3

Preguntas del Examen de Simulación 2

#	Pregunta	K
183.	<p>Objetivo del aprendizaje: LO-5.1.3</p> <p>Consideré los siguientes miembros de un equipo de proyecto:</p> <ul style="list-style-type: none"> I. Un jefe de proyecto. II. Un usuario. III. Un operador. IV. Un jefe de desarrollo. V. Un desarrollador. <p>¿Cuál de las siguientes afirmaciones es la más verdadera acerca de los miembros del equipo en el rol de probadores?</p> <ul style="list-style-type: none"> A. Es posible que todos los cinco miembros sean probadores durante el proyecto. B. Es posible que I y IV sean probadores durante el proyecto. C. Es posible que II, III y V sean probadores durante el proyecto. D. Es posible que ninguno sea probador, porque sólo los probadores profesionales deberían trabajar en un rol de pruebas. 	1
184.	<p>Objetivo del aprendizaje: LO-5.2.8</p> <p>Usted está comprometido con la planificación del esfuerzo de las pruebas para una nueva aplicación bancaria en línea. Para averiguar cuánto esfuerzo, tiempo y recursos se requiere, primero, usted se reúne con el equipo de pruebas propuesto y con los otros en el proyecto. Para verificar los resultados de la estimación, Usted se basa en algunos promedios de la industria para el esfuerzo y los costos de las pruebas en proyectos similares, publicados por un consultor de renombre. ¿Cuál afirmación describe de manera exacta su método de estimación?</p> <ul style="list-style-type: none"> A. Un método simultáneo basado en los expertos y las métricas. B. Principalmente un método basado en los expertos, ampliado con un método basado en las métricas. C. Principalmente un método basado en las métricas, ampliado con un método basado en los expertos. D. Un método completamente ascendente. 	2
185.	<p>Objetivo del aprendizaje: LO-5.6.2</p> <p>Consideré el siguiente texto de un informe de defecto relacionado al rendimiento:</p> <p>Resumen: La utilización del sistema es demasiada alta con 50 usuarios.</p> <p>Descripción de la incidencia:</p> <ol style="list-style-type: none"> 1. Ponga en marcha el generador de carga e incremente la carga a 50 usuarios simultáneos. 2. La utilización del CPU aumenta al 55%. La utilización de la memoria aumenta al 32%. <p>Impacto: El sistema se vuelve lento bajo carga.</p> <p>¿Cuál de las siguientes afirmaciones resumen mejor cualquier problema que podría existir con este informe de defecto?</p> <ul style="list-style-type: none"> A. Está bien como está. B. No es neutral y ni enfocado en los hechos. C. No especifica los resultados esperados. D. No especifica los resultados reales. 	3
186.	<p>Objetivo del aprendizaje: LO-5.3.1</p> <p>¿Cuál de las siguientes es una métrica de prueba común, utilizada a menudo para monitorear ambos, la preparación de las pruebas y la ejecución de pruebas?</p> <ul style="list-style-type: none"> A. El estado de los casos de prueba. B. Las tasas de hallazgos/corrección de los defectos. C. La confianza subjetiva de los probadores en el producto. D. El costo estimado para encontrar el siguiente defecto. 	1
187.	<p>Objetivo del aprendizaje: LO-5.6.2</p> <p>Usted está trabajando como un probador en un sistema bancario en línea. La disponibilidad es considerada como uno de los riesgos más altos (de calidad) del producto para el sistema. Usted encuentra una falla reproducible que resulta en la pérdida de la conexión de los clientes al sitio Web del banco cuando fondos son transferidos entre tipos comunes de cuentas y no pueden reconectarse entre 3 a 5 minutos.</p>	3

	<p>¿Cuál de los siguientes sería un buen resumen acerca de un informe de incidencia para esta falla, uno que capture ambos la esencia de la falla y su impacto en los interesados del negocio?</p> <ul style="list-style-type: none"> A. Los registros del servidor Web muestran el error 0x44AB27 cuando se está ejecutando la prueba 07.005, lo cual no es un mensaje de error esperado en el directorio /tmp. B. Los desarrolladores han introducido un defecto de disponibilidad importante, el cual molestará seriamente a nuestros clientes. C. El rendimiento es lento y la fiabilidad es inestable bajo carga. D. Una transacción típica de la transferencia de fondos resulta en la terminación de la sesión del cliente, con un retraso en la disponibilidad cuando intentan reconectarse. 	
188.	<p>Objetivo del aprendizaje: LO-5.4.1</p> <p>Usted está trabajando como un jefe de proyecto en un proyecto interno de software bancario. Para prevenir el reproceso y los ciclos excesivos de encontrar/corregir/repetir las pruebas, el siguiente proceso ha sido puesto en vigencia para resolver un defecto una vez que haya sido encontrado en el laboratorio de pruebas:</p> <ol style="list-style-type: none"> 1. El desarrollador asignado encuentra y corrige el defecto, luego crea una compilación experimental. 2. Un colega desarrollador revisa, realiza las pruebas unitarias y las pruebas de confirmación de la corrección del defecto en su computador de escritorio. 3. Un probador — usualmente el que encontró el defecto — realiza las pruebas de confirmación de la corrección del defecto en el entorno del desarrollo. 4. Una vez a la semana, una nueva versión incluida con todas las correcciones de defectos, es instalada en el laboratorio de pruebas. 5. El mismo probador del paso 3 realiza las pruebas de confirmación de la corrección del defecto en el entorno de pruebas. <p>Sin embargo, un gran número de defectos, los cuales los probadores los confirmaron como corregidos en el entorno de desarrollo, están fallando de alguna manera en las pruebas de confirmación en el entorno de pruebas, resultando en el reproceso y más ciclos de tiempo. Usted tiene la más alta confianza en su equipo de pruebas y ha descartado equivocaciones u omisiones en el paso 3 de arriba.</p> <p>¿Cuál de las siguientes es la parte más probable del proceso que debe ser comprobada la próxima vez?</p> <ul style="list-style-type: none"> A. Los desarrolladores quienes pueden no estar probando correctamente en el paso 2. B. Los desarrolladores quienes pueden estar confundidos acerca de qué probar en el paso 5. C. La gestión de configuración, la cual puede que no esté manteniendo la integridad del producto en el paso 4. D. Los desarrolladores, quienes pueden no estar corrigiendo los defectos correctamente en el paso 1. 	2
189.	<p>Objetivo del aprendizaje: LO-5.5.2</p> <p>¿Cuál de los siguientes son dos factores que determinan el nivel de riesgo?</p> <ul style="list-style-type: none"> A. Las pruebas y el desarrollo. B. Lo dinámico y reactivo. C. La sentencia y decisión. D. La probabilidad y el impacto. 	1
190.	<p>Objetivo del aprendizaje: LO-5.6.2</p> <p>Considere las siguientes afirmaciones de un informe de incidencia:</p> <p>El sistema es completamente incapaz de procesar cualquier pago en línea, que usa tarjetas de American Express; esas transacciones son el 35% de nuestras ventas.</p> <p>¿A cuál de los siguientes detalles típicos de un informe de incidencia se relaciona esta frase más estrechamente?</p> <ul style="list-style-type: none"> A. La fecha cuando la incidencia fue descubierta. B. El ítem de configuración del software. C. El grado del impacto. D. El estado de la incidencia. 	2

29 Toolsmith es una persona que construye herramientas, con las cuales otros ingenieros de software construyen o prueban programas.

³⁰ El proyecto Sumatra es acerca de adicionar un Sistema de Gestión de Documentos (DMS) al producto SpeedyWriter-procesador web de texto.

Capítulo 6

Soporte de Herramientas para las Pruebas

"La ciencia no es perfecta, con frecuencia se utiliza mal, no es más que una herramienta, pero es la mejor herramienta que tenemos, se corrige a sí misma, está siempre evolucionando y se puede aplicar a todo. Con esta herramienta conquistamos lo imposible"

Carl Edward Sagan, astrónomo, exobiólogo y divulgador científico.

El capítulo 6, Soporte de Herramientas para las Pruebas, contiene las siguientes 3 secciones:

1. Tipos de herramientas de pruebas.
2. Uso efectivo de herramientas, los beneficios y los riesgos potenciales.
3. Introducción de una herramienta en una organización.

Cada una de las secciones estará desglosada en dos o más partes.

6.1 Tipos de Herramientas de Pruebas

Objetivos del Aprendizaje

LO-6.1.1 Clasificar los diferentes tipos de herramientas de pruebas de acuerdo con sus propósitos y las actividades del proceso de pruebas básico y el ciclo de vida de software. (K2)

LO-6.1.3 Explicar el término herramienta de pruebas y el propósito del soporte de las herramientas para las pruebas. (K2)

Esta sección, Tipos de Herramientas de Pruebas, cubrirá los siguientes conceptos clave:

- Diferentes tipos de herramientas de pruebas.
- Tipos de herramientas utilizadas por los programadores.

Las herramientas de pruebas apoyan varias actividades que ocurren durante las pruebas.

Algunas herramientas son utilizadas directamente. Por ejemplo, las herramientas de ejecución de pruebas que ejecutan las pruebas predefinidas o automáticamente generadas, las herramientas de generación de datos de prueba que crean los datos utilizados por las pruebas ya sea a través de la entrada directa o a través del llenado previo de la base de datos y las herramientas de comparación de resultados que buscan anomalías en los resultados reales de las pruebas.

Algunas herramientas realizan el monitoreo durante las pruebas. Por ejemplo, algunas herramientas monitorean la memoria y otras monitorean la actividad de los archivos. Esto es relacionado algunas veces como el reconocimiento o la exploración.

Algunas herramientas son simplemente una ayuda para las pruebas. La hoja de cálculo es el ejemplo más común de una herramienta de pruebas en este sentido.

Las herramientas sirven para varios propósitos durante las pruebas, en algunos casos para múltiples propósitos. Esto incluye lo siguiente:

Las herramientas pueden mejorar la eficiencia de las pruebas. Un ejemplo es la automatización de las tareas repetitivas como las pruebas de regresión. Otro ejemplo es la simplificación y (en alguna extensión) la automatización de las actividades manuales de las pruebas como la planificación de las pruebas, el diseño de las pruebas, la creación de los informes y el monitoreo de las pruebas.

Las herramientas pueden automatizar las actividades que requerirían mucho tiempo, atención, esfuerzo y recursos si las personas tuvieran que realizarlas manualmente. Un ejemplo de esto es la comprobación de las violaciones de los estándares de codificación utilizando una herramienta de análisis estático.

Glosario del ISTQB

Herramienta de análisis estático: Véase analizador estático.

Analizador estático: Una herramienta que lleva a cabo el análisis estático. Note que este término no es enunciado específicamente para esta sección pero está incluido aquí porque es un sinónimo de herramienta de análisis estático.

Las herramientas pueden automatizar las actividades que simplemente no pueden ser realizadas manualmente. Los ejemplos más claros aquí son las pruebas de fiabilidad y rendimiento, especialmente para las aplicaciones cliente-servidor, del internet y la nube.

Las herramientas pueden incrementar la fiabilidad de las pruebas. Por ejemplo, las herramientas pueden automatizar grandes comparaciones de datos cuando los archivos o las tablas de la base de datos están involucrados en las pruebas y las herramientas pueden simular los comportamientos como los flujos de entradas.

Hay un gran número de herramientas de pruebas disponibles, lo cual es tanto una bendición como una maldición, por razones que usted observará a la medida que continúe a través de este capítulo.

Para ayudar a comprender el rango del soporte de las herramientas de pruebas, en el programa de estudios Nivel Básico, clasificamos y tratamos las herramientas de pruebas en base a las principales actividades de las pruebas, que son cubiertas. En algunos casos, las herramientas tienen la meta claramente en una sola actividad, pero algunas herramientas apoyan múltiples actividades. Es posible que encuentre que algunas herramientas de algunos proveedores agrupadas en un paquete, que apoya un rango de actividades.

Es posible que encuentre gente que clasifica o se refiere a las herramientas en base al propósito de la herramienta, la tecnología utilizada y la forma de adquisición de la herramienta. Sin embargo, en cuanto a las formas adquisición de las herramientas encontrará a menudo opciones disponibles como comerciales, libres, de código abierto y shareware.

Las herramientas de pruebas son ya sea intrusivas o no intrusivas, dependiendo de que si la herramienta se ejecuta en un sistema sometido a pruebas. Si la herramienta es ejecutada en el sistema sometido a pruebas, ésta puede afectar el resultado de la prueba. Esta propiedad de las herramientas intrusivas es denominada el efecto sonda. Las herramientas no intrusivas, mientras no tienen un efecto sonda, son generalmente más caras.

Algunas herramientas son principalmente utilizadas por los desarrolladores, y eso usted observará mientras va a través de la clasificación de las herramientas en los párrafos siguientes.

Una palabra popular y que está de moda acerca de las herramientas de pruebas es el término "framework de pruebas". Es posible que escuche esto de los expertos en la automatización de pruebas y la literatura del marketing de los proveedores de herramientas. Como muchas palabras famosas, ésta ha sido sobrecargada con significados. Entonces, algunas veces la gente utiliza "framework de pruebas" para querer decir una biblioteca reutilizable y extensible de pruebas que los probadores pueden emplear para construir herramientas sofisticadas de pruebas, e, imprecisamente, este significado es también referido como "arnés de pruebas". Un "framework de pruebas" puede significar un tipo de método del diseño de la automatización de las pruebas, así como las pruebas dirigidas por los datos y por las palabras clave. Finalmente, hay el significado más amplio del "framework de pruebas", el cuál es el proceso general de la ejecución de las pruebas. En este libro y el programa de estudios Nivel Básico, el término "framework de pruebas" significa ya sea una biblioteca reutilizable, extensible o un método del diseño de la automatización de las pruebas.

Glosario del ISTQB

Arnés de pruebas: Un entorno de pruebas compuesto de stubs y drivers necesarios para ejecutar una prueba.

Las herramientas de gestión de pruebas tienen las siguientes funciones:

- Proporcionar la trazabilidad de las pruebas, los resultados y las incidencias de las pruebas a la base de las pruebas.
- Permitir el registro de los resultados de las pruebas y la generación de los informes del progreso.
- Ayudar a gestionar las pruebas y el proceso de pruebas.
- Interactuar con la ejecución de las pruebas, el seguimiento de los defectos y las herramientas de gestión de requisitos.
- Proporcionar el control de las versiones o la interfaz con una herramienta de gestión de configuración.
- Realizar el análisis cuantitativo (o las métricas) relacionados con la pruebas.

Las herramientas de gestión de requisitos tienen las siguientes funciones:

- Guardar las declaraciones de los requisitos.
- Comprobar la coherencia y los requisitos indefinidos (o faltantes).
- Permitir la priorización de los requisitos.
- Hacer posible que las pruebas individuales sean trazables (e informadas en términos de) a los requisitos, las funciones, las características, y, en algunos casos, a los riesgos.

Las herramientas de gestión de "bugs" o incidencias o defectos tienen las siguientes funciones:

- Guardar y gestionar los informes de los defectos.
- Facilitar la priorización y la clasificación de los defectos.
- Proporcionar un flujo del trabajo basado en los estados, incluyendo la asignación de acciones a la gente. Por ejemplo, corregir un defecto, la prueba de confirmación de lo corregido y etc.
- Hacer posible el monitoreo de los defectos de un proyecto y del estado de un defecto con el tiempo, especialmente en relación con las tendencias.
- Proporcionar el soporte para el análisis estadístico (muchas veces a través de la exportación, por ejemplo por medio de una herramienta como Excel).
- Crear informes y gráficas (aunque estos son a menudo de utilidad limitada debido a la variación de las necesidades).

Las herramientas de gestión de configuración tienen las siguientes funciones:

- Guardar la información acerca de las versiones y las construcciones del software y testware.
- Hacer posible la trazabilidad entre el testware y los productos del trabajo del software y las

variantes del producto.

- Ayudar con el desarrollo y las pruebas en múltiples configuraciones de los entornos de hardware/software.

Las herramientas de soporte del proceso de revisión tienen las siguientes funciones:

- Guardar la información acerca de los procesos de revisión.
- Guardar y comunicar los comentarios de las revisiones
- Informar acerca de los defectos y el esfuerzo.
- Gestionar las referencias a las reglas de revisión y/o listas de comprobación.
- Proporcionar ayuda para las revisiones en línea.
- Seguir la trazabilidad entre los documentos y el código fuente.

Las herramientas del análisis estático son principalmente utilizadas por los desarrolladores y tienen las siguientes funciones:

- Encontrar los defectos antes de las pruebas dinámicas.
- Hacer valer los estándares de código.
- Analizar las estructuras y dependencias.
- Ayudar en la comprensión del código fuente.
- Calcular las métricas del código.

Las herramientas de modelado y diseño son principalmente utilizadas por los desarrolladores y tienen las siguientes funciones:

- Ayudar a crear modelos del sistema.
- Validar los modelos.
- Ayudar en la generación de algunos casos de prueba basados en el modelo.

Las herramientas de diseño de pruebas tienen las siguientes funciones:

- Generar las entradas de las pruebas o las pruebas reales de los requisitos, de la interfaz gráfica del usuario, de los modelos del diseño y del código.
- Generar los resultados esperados (aunque la fiabilidad de tales oráculos de pruebas es a menudo limitada).
- Generar frameworks, plantillas y stubs de pruebas.

Glosario del ISTQB

Herramienta de diseño de pruebas: Una herramienta que apoya la actividad del diseño de pruebas por medio de la generación de entradas de pruebas de una especificación que puede ser contenida en un repositorio de una herramienta CASE, p.ej. una herramienta de gestión de requisitos, de las condiciones de pruebas especificadas contenidas en la misma herramienta o del código.

Las herramientas de datos de prueba tienen las siguientes funciones:

- Manipular o crear bases de datos, archivos o datos para la utilización durante la ejecución de las pruebas.
- Crear grandes volúmenes de datos útiles de prueba.
- Validar los datos de prueba según reglas específicas.
- Analizar los datos acerca de la frecuencia de las condiciones, y así sucesivamente.
- Mezclar o anonimizar los datos de los clientes en tiempo real.

Las herramientas de ejecución de pruebas, en las que la mayoría de las personas piensan cuando estas consideran las herramientas de pruebas, tienen las siguientes funciones:

Ejecutan las pruebas, en otras palabras, envían las entradas para el guión automatizado y comparan los resultados reales con los esperados.

Correctamente utilizado, el guión es un procedimiento de prueba de entrada independiente, que permite repetir las pruebas con datos diferentes, ejecutar pruebas similares y mantener fácilmente las pruebas.

Incorrectamente utilizado—como puede ocurrir con la típica reproducción y captura— los guiones están entrelazados con los datos y/o los resultados esperados, tienen retrasos en código duro en estos y son bastante frágiles.

La reproducción y captura pueden ser útiles durante las pruebas exploratorias para grabar las pruebas que ocurrieron.

La mayoría de las herramientas de ejecución de pruebas incluyen comparadores. A menudo producen registros analizables. Pueden trabajar en la interfaz gráfica del usuario, en la interfaz de programación de aplicaciones (“API”) o la interfaz de línea de comando.

Los arneses de pruebas, los frameworks y los simuladores tienen las siguientes funciones:

- Reemplazar o sustituir las partes faltantes y potencialmente problemáticas del hardware o software.
- Facilitar las pruebas de una unidad o unidades por medio de la generación y/o apoyo de los drivers, stubs y/u los objetos de simulación que reemplazan partes del sistema, que no están disponibles o han sido extraídas para aislar la unidad.

- Proporcionar los frameworks de ejecución en el middleware para probar los lenguajes, los sistemas operativos o el hardware.

Los comparadores de pruebas tienen las siguientes funciones:

- Comprobar los archivos, las bases de datos o los resultados de las pruebas contra las expectativas, ya sea durante la ejecución de las pruebas o después.
- Estos pueden incluir un oráculo automatizado de pruebas en vez de la comparación contra líneas bases estáticas.
- Se los puede hacer más inteligentes programándolos para manejar las variables dinámicas como las fechas y los horarios o para ignorar el orden de los registros cuando el orden de la posición en los informes o consultas es ambiguo.

Glosario del ISTQB

Comparador de pruebas: Una herramienta de pruebas para realizar la comparación automatizada de las pruebas de los resultados reales con los resultados esperados.

Las herramientas de medición de la cobertura de código son utilizadas por los desarrolladores. Éstas son ya sea intrusivas o no intrusivas. Éstas miden el porcentaje de los tipos específicos de estructura del código que han sido ejercidos. Como se mencionó en el capítulo 4, éstas pueden medir la cobertura del código en cuanto a las sentencias, las ramas o las decisiones, los objetos, las invocaciones de funciones u otros elementos estructurales.

Estas herramientas comprueban qué tan rigurosamente un conjunto de pruebas ha ejecutado el tipo medido de estructura.

Las herramientas de seguridad tienen las siguientes funciones:

- Comprobar los virus de las computadoras.
- Simular varios tipos de ataques.
- Simular varias condiciones de seguridad.
- Comprobar el código acerca de las violaciones conocidas de seguridad.

Glosario del ISTQB

Herramienta de seguridad: Una herramienta que apoya la seguridad operacional.

Herramienta de pruebas de seguridad: Una herramienta que brinda el soporte para probar las características y las vulnerabilidades de seguridad. Note que este término no es enunciado específicamente en esta sección, pero está incluido aquí para ayudarle a comprender el término herramienta de seguridad.

Las herramientas de rendimiento, monitoreo y análisis dinámico son principalmente utilizadas por los desarrolladores, aunque muchos probadores utilizan las herramientas de rendimiento y monitoreo.

Las herramientas de análisis dinámico son a menudo utilizadas para comprobar las dependencias de tiempo o las fugas de memoria.

Éstas pueden monitorear e informar acerca de cómo un sistema se comporta con condiciones simuladas de utilización. También pueden generar varias condiciones de carga (con una esperanza realista) para la aplicación, una base de datos, red o servidor, a menudo por algún guión o procedimiento programado. Pueden monitorear, analizar, verificar e informar la utilización de recursos específicos del sistema y dar advertencias acerca de posibles problemas.

A menudo otras herramientas son utilizadas para las pruebas.

Algunas herramientas están enfocadas en aplicaciones particulares.

Hay herramientas de pruebas de rendimiento basadas en la web.

Hay herramientas de análisis estático específicas para un lenguaje.

Hay herramientas de pruebas de seguridad.

Algunas herramientas de pruebas tienen como objetivo áreas específicas de aplicaciones como los sistemas embebidos.

Por supuesto, los probadores también utilizan las hojas de cálculo (¡muchas de ellas!) y bases de datos.

Muchos desarrolladores utilizan herramientas de depuración.

Algunas herramientas están enfocadas en la evaluación de la calidad de los datos. Para proyectos que implican la conversión de los datos, la migración de los datos y los almacenes de datos, la calidad de los datos es una consideración esencial, tanto en cuanto a la criticidad como al volumen. Para estos proyectos, los probadores necesitan herramientas para evaluar la calidad de los datos, para revisar y verificar la conversión y migración de los datos, para asegurar el procesamiento correcto y completo de potencialmente volúmenes gigantes de datos, y para verificar la conformidad con cualquier estándar aplicable.

6.1.1 Ejercicios

Ejercicio 1

Haga una lista de los tipos de herramientas abordados en esta sección, los cuales piensa usted que serían útiles para Omnitnet.

Argumente.

Solución del Ejercicio 1

Muchas de las herramientas abordadas en la sección previa serían útiles para Omnitnet:

- Herramientas de gestión de pruebas: una hoja de cálculo del seguimiento de las pruebas, una base de datos que ayuda en la gestión de la complejidad del entorno y de las plantillas para los informes.
- Herramientas de gestión de requisitos: realizan el seguimiento de la trazabilidad de los requisitos a los riesgos y las pruebas.
- Herramientas de gestión de incidencias: una base de datos del seguimiento de defectos, libre o comercial.
- Herramientas de gestión de configuración: repositorio de fuentes con la capacidad del control de cambios.
- Herramientas de análisis estático: comprueban la gramática de los requisitos y modelan el rendimiento del centro de datos.
- Herramientas de diseño de pruebas: generan las tablas de las pruebas de configuración con todos los pares de variables presentes.
- Herramientas de preparación de datos de prueba: crean datos realistas de los clientes para las pruebas del centro de datos y centro de llamadas.
- Herramientas de ejecución de pruebas: automatizan las pruebas de regresión para el centro de llamadas y posiblemente el quiosco.
- Arneses de pruebas y frameworks de pruebas unitarias: utilizan herramientas libres o comerciales para todas las pruebas unitarias del código y posiblemente las pruebas de integración y la creación de generadores de carga.
- Comparadores de pruebas: incluidas en la mayoría de las herramientas de ejecución de pruebas.
- Herramientas de medición de cobertura: comprueban las pruebas del código crítico del quiosco, el centro de llamadas y el centro de datos.
- Herramientas de seguridad: pruebas de penetración.
- Herramientas de análisis dinámico: monitorean los niveles de utilización de los quioscos, el centro de datos y el centro de llamadas durante las pruebas.
- Herramientas de pruebas de rendimiento, carga y estrés: automatizan las pruebas de rendimiento, carga y estrés para el centro de datos.

Puede encontrar muchas de estas herramientas disponibles de forma libre o usted mismo puede construirlas rápidamente utilizando aplicaciones estándar de oficina como Excel y Access.³¹

6.2 Uso efectivo de Herramientas, los Beneficios y los Riesgos Potenciales

Objetivos del Aprendizaje

LO-6.2.1 Resumir los beneficios y los riesgos potenciales de la automatización de pruebas y el soporte de herramientas para las pruebas. (K2)

LO-6.2.2 Recordar las consideraciones especiales para las herramientas de ejecución de pruebas, el análisis estático y la gestión de pruebas. (K1)

Glosario del ISTQB

(Pruebas) Dirigidas por datos (“data driven”): Una técnica de guión que guarda las entradas de las pruebas y los resultados esperados en una tabla o una hoja de cálculo, de tal forma que un sólo guión de control pueda ejecutar todas las pruebas en la tabla. Las pruebas dirigidas por datos son a menudo utilizadas para apoyar la aplicación de las herramientas de ejecución de pruebas así como las herramientas de captura/reproducción. [Fewster and Graham] Véase también las pruebas dirigidas por palabras clave.

(Pruebas) Dirigidas por palabras clave (“key word driven”): Una técnica de guión que utiliza archivos de datos para contener no sólo datos de prueba y resultados esperados, sino también palabras clave relacionadas con la aplicación que está siendo probada. Las palabras clave son interpretadas por guiones especiales de apoyo que son invocados por el guión de control para la prueba. Véase también pruebas dirigidas por datos.

Lenguaje de guión: Un lenguaje de programación en el cual los guiones ejecutables de las pruebas son escritos y utilizados por una herramienta de ejecución de pruebas (p.ej. Una herramienta de captura/reproducción)

Esta sección, Uso efectivo de herramientas, los beneficios y los riesgos potenciales, cubrirá los siguientes conceptos clave:

- Diferentes técnicas de guiones para las herramientas de ejecución de pruebas, incluyendo los beneficios y los riesgos potenciales de la automatización de las pruebas dirigida por datos y dirigida por palabras clave.

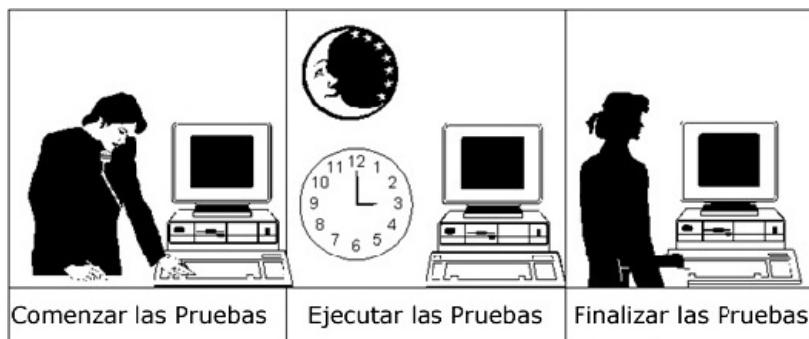
La automatización de pruebas presenta varias oportunidades—y riesgos.

Las oportunidades incluyen lo siguiente:

- Posibilidad reducida de rehacer los trabajos. Hay pocas cosas más pesadas que las pruebas manuales de regresión una vez que la prueba ha sido ejecutada dos o tres veces. Este tedioso trabajo conduce a equivocaciones.
- Coherencia y repetibilidad mejorada. Porque la gente tiende a cometer errores cuando ejecuta las pruebas nuevamente o cuando trata de crear condiciones precisas de pruebas cada vez, las herramientas pueden ayudarle a ejecutar exactamente las mismas pruebas.
- Evaluación objetiva. Los criterios de medición son precisos para una prueba automatizada—Estos deben ser así. Esto es verdad especialmente para las pruebas de cobertura, rendimiento y fiabilidad.
- Informes simplificados. Las pruebas automatizadas recolectan los registros estandarizados que fácilmente pueden ser extraídos, analizados y manipulados.

Los riesgos incluyen lo siguiente:

- Expectativas poco realistas: Éste es el más grande. La gente espera que las herramientas de automatización de pruebas les resuelvan todos sus problemas, incluyendo las cuestiones del tiempo y dinero. Eso no sucederá.
- Subestimación del tiempo, el costo y el esfuerzo del desarrollo, la ejecución y el mantenimiento. Es costoso llegar a un punto de automatización significativa de las pruebas. La mayoría de las organizaciones que tratan de hacerlo así, tienen muchos problemas para lograr eso. Muchas de estas organizaciones claudican antes de llegar allí porque no presupuestaron lo suficiente para eso. En otros casos, la introducción inicial es alcanzada, pero los costos de ejecución y mantenimiento de las pruebas son subestimados a largo plazo, conduciendo al abandono de las pruebas automatizadas. Aún otra posibilidad es que las implicaciones de los cambios en el proceso de pruebas y la necesidad para la mejora continua en la utilización de la herramienta son subestimados, lo cual puede conducir ya sea al abandono o a retornos por debajo de lo óptimo de la utilización de las pruebas automatizadas.
- Utilización de la herramienta para pruebas inadecuadas. Algunas pruebas no son simplemente automatizables. Algunas podrían ser realizadas mejor manualmente. Algunos probadores consideran las herramientas de pruebas automatizadas como un substituto para un buen diseño de pruebas, lo cual no es la manera correcta de pensar acerca de las herramientas de pruebas automatizadas.
- Gestión incorrecta del control de versión y configuración de los varios productos del trabajo. Mientras que puede ser complicado y difícil de gestionar un gran conjunto de productos del trabajo para los casos de prueba manuales, las pruebas automatizadas son típicamente más difíciles. Básicamente, las pruebas automatizadas son una forma de software, entonces todas las mejores prácticas de la gestión de configuración tratadas en el capítulo 5 aplican.
- Fracaso en el control de las cuestiones de relaciones e interoperabilidad entre las herramientas críticas. Un conjunto completo de herramientas de pruebas, pueden incluir las herramientas de gestión de requisitos, las herramientas de control de versión y las herramientas de seguimiento de defectos, posiblemente de múltiples proveedores. Puede ser complicado de hacer funcionar estas herramientas juntas.
- La herramienta huérfana o la herramienta descuidada. El proveedor de herramientas podría terminar el negocio, retirar la herramienta, o vender la herramienta a otro proveedor. Alternativamente, el proveedor de la herramienta simplemente podría ignorar la herramienta, dar un soporte inadecuado, pocas e infrecuentes actualizaciones y pocas o nada de correcciones de defectos. Los mismos riesgos aplican para las herramientas de código abierto, si la comunidad alrededor de la herramienta se disuelve o se vuelve inactiva.
- El riesgo del lado ciego. Cuestiones imprevistas pueden surgir totalmente, como la incapacidad de ser compatible con una nueva plataforma.



Si usted se enfoca sólo en la ejecución de las pruebas, la automatización de las pruebas parece muy eficiente, pero no se olvide que las pruebas automatizadas tuvieron que ser *automatizadas por alguien* para ser automatizadas



Figura 6.1: Teoría y Práctica de las Pruebas Automatizadas

Hay un gran vacío entre la teoría de la automatización de pruebas—o los reclamos de marketing—y la práctica de la automatización de pruebas.

Por un lado, no importa lo que cualquier folleto ilustrado o vendedor amable le diga, recuerde que la automatización de pruebas es normalmente desarrollo de software. Usted está programando una computadora para que se pruebe así misma o pruebe otra computadora. Ése es un proceso no trivial que requiere mucho tiempo, habilidad y dinero.

Ahora, si invierte en la automatización sabiamente, tendrá su dinero de vuelta—tendrá un retorno positivo de la inversión—pero usualmente no en el proyecto piloto. Usualmente, toma meses o aún años para ahorrar la misma cantidad de tiempo y dinero que invirtió en la automatización.

Hay algunas excepciones en estas reglas. Cuando utilice frameworks de pruebas automatizadas como x-Unit para realizar pruebas simples de unidad, componente e integración basadas en un API, el esfuerzo es pequeño y los beneficios son inmediatos. También, algunas pruebas no funcionales como las pruebas de fiabilidad y rendimiento son simples de crear y producen los beneficios inmediatamente.

Otro vacío emerge del nombre. Cuando piensa en “automatización de pruebas” puede que piense en la automatización del proceso de pruebas entero, desde la planificación pasando por el análisis, el diseño, la implementación, la ejecución, el control hasta el cierre. Sin embargo, lo que usualmente se automatiza es solamente la ejecución de las pruebas. De hecho, es usualmente sólo una minúscula parte del proceso de la ejecución de pruebas, el envío de las entradas y los eventos al sistema, la comparación de los resultados esperados con los reales, y el registro de esa comparación. Todo el trabajo previo—el análisis, el diseño y la implementación—no sólo es no automatizado, pero de hecho es la importante labor que se necesita **para** automatizar. El trabajo posterior, especialmente el análisis de los resultados no es automatizado. En realidad, la automatización, incorrectamente realizada, puede complicar el análisis de las pruebas falladas y aumentar el número de pruebas falladas.

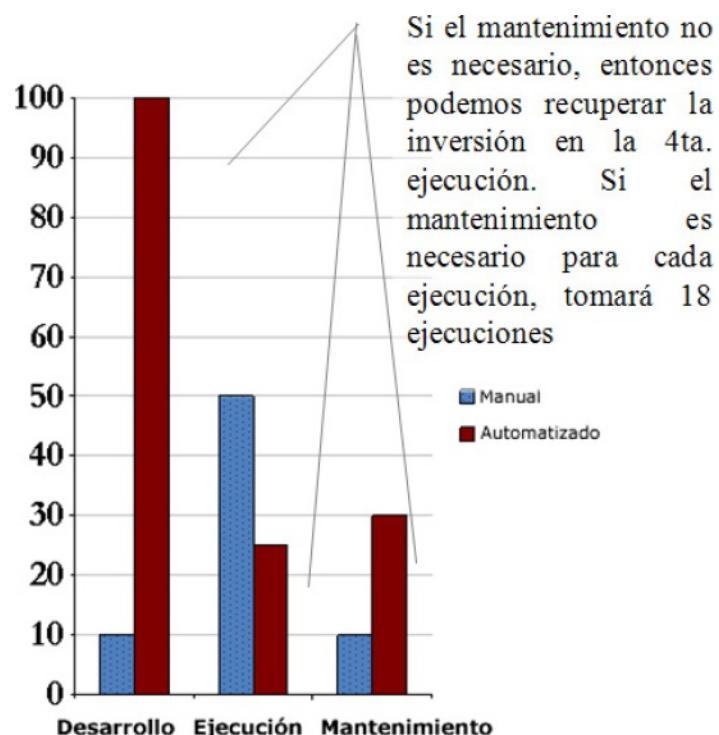


Figura 6.2: Retorno de Inversión (ROI) de la Automatización de Pruebas

Normalmente, una motivación típica de la organización para invertir el dinero en la automatización de pruebas es la obtención de un retorno de esa inversión en cuanto al esfuerzo para obtener un nivel dado de mitigación de los riesgos.

En esta figura, usted puede observar un ejemplo hipotético que compara el costo promedio del desarrollo, la ejecución y el mantenimiento de las pruebas automatizadas versus las manuales. Pase un momento estudiando la gráfica.

Puede ver lo que cuesta, en promedio, US\$ 100 por prueba para desarrollar las pruebas automatizadas, mientras que las pruebas manuales cuestan sólo 10US\$. Entonces, usted invierte US\$ 90 por prueba en promedio para automatizar.

Inmediatamente, podemos ver un camino para el retorno de la inversión de la automatización mejorada. Como puede imaginar, hay altos costos fijos que tienen que ver con la automatización, incluyendo la compra de la herramienta, la configuración del entorno, la creación del framework de pruebas, y así sucesivamente. Estos costos son constantes—y altos—sin importar si queremos automatizar una o mil pruebas. Mientras más pruebas queremos automatizar, más pruebas podemos extender sobre aquellos costos fijos. Así que para un mejor ROI de la automatización de pruebas, deberíamos tratar de automatizar cuando hay muchas pruebas **para** automatizar.

Note que el costo de la ejecución de las prueba es menor para las pruebas automatizadas en este ejemplo. La diferencia entre los costos de la ejecución de las pruebas automatizadas y manuales es de donde el ROI se obtendrá. Ahorramos US\$ 25 cada vez que ejecutamos una prueba automatizada, en promedio, en este caso de estudio hipotético. Entonces, si ejecutamos la prueba cuatro veces, tenemos de vuelta nuestros US\$ 90, mas US\$ 10 de dividendos.

Una vez más, cuanto más grande el número de pruebas, mejor será la tendencia de los beneficios. Eso es porque la ejecución de las pruebas automatizadas incluye un componente de costo fijo bastante grande asociado con la instalación del entorno y los datos y el lanzamiento de las pruebas. Cuantas más pruebas tengamos que ejecutar, más pruebas podemos extender a través de **estos** costos fijos.

Ahora, aquí viene el elemento que mucha gente olvida—para su gran perjuicio—cuando se automatizan las pruebas. Las pruebas automatizadas deben ser mantenidas a medida que el sistema evoluciona. Note en nuestro ejemplo, que gastamos US\$ 20 más por prueba para el mantenimiento de la prueba automatizada.

Si esto solo ocurre una vez cada 12 o 18 meses por prueba, en promedio, no es gran cosa. Pero, ¿Supone usted que ocurre cada vez que tenemos que ejecutar las pruebas? En ese caso, tomaría **dieciocho** ejecuciones de pruebas para alcanzar el equilibrio. Esas son muchas ejecuciones de pruebas. Podría tomar años para obtener un ROI positivo.

En algunos casos, esto se pone tan difícil que la organización tiene un ROI negativo. En otras palabras, los costos actuales del mantenimiento devoran más dinero que lo que es ahorrado por la ejecución. Entonces, considere la estabilidad del sistema que intenta probar con las herramientas automatizadas. Si está cambiando muy rápidamente, no tendrá un ROI positivo a causa de los costos de mantenimiento. También, aún para los sistemas de moderado a lento cambio, usted también necesita un framework de pruebas bien diseñado y mantenable. Es por esto que necesita probadores hábiles involucrados en la automatización de pruebas.

Aún si tiene un ahorro neto del esfuerzo, recuerde que las pruebas no tienen valor por sí mismas. Las pruebas sólo tienen valor cuando ayudan a la organización a alcanzar un objetivo valioso. Si está automatizando las pruebas de regresión, entonces debe haber suficientes riesgos altos de regresión para justificar la repetición de esas pruebas. Simplemente porque **puede** automatizar y repetir las pruebas una y otra vez con un ahorro neto de esfuerzos **no significa** que usted debería automatizar. Las herramientas de pruebas son solo eso—herramientas—y la automatización de pruebas es una técnica. Las herramientas de pruebas y la automatización de pruebas no son estrategias de pruebas, aunque mucha gente las confunde por tales.

Para alcanzar un retorno de inversión positivo de las pruebas automatizadas, tenemos que automatizar las pruebas que pueden ser automatizadas.

Generalmente, cualquier prueba que requiera la frecuente intervención humana para mantenerla en ejecución, o el juicio humano para interpretar sus resultados, es la prueba más apropiada para las pruebas manuales. Esas pruebas incluyen típicamente lo siguiente:

- Operaciones y mantenimiento.
- Configuración y compatibilidad.
- Control de errores y recuperación.
- Localización.
- Usabilidad.
- Instalación y ajustes.
- Documentación y ayuda.

De nuevo, generalmente, cualquier prueba que requiera precisión exquisita o la cuál deba ser repetida muchas veces, esa es la prueba más apropiada para las pruebas automatizadas. Esas pruebas incluyen típicamente lo siguiente:

- Regresión y confirmación.
- Pruebas de mono (o aleatorias).
- Carga, volumen y capacidad.
- Rendimiento y fiabilidad.
- Cumplimiento de estándares.
- Caja blanca, especialmente de unidad, componente e integración basadas en un API.
- Complejidad estática y análisis de código.

Algunas pruebas pueden ser realizadas utilizando técnicas manuales, automatizadas o combinadas, así como:

- Funcionales.
- Casos de uso o escenarios de usuario.
- Interfaz de usuario.
- Control de fechas y horarios.

Pruebas manuales inapropiadas engañan a la gente acerca de la cobertura de pruebas. Por ejemplo, observamos a alguien realizando pruebas de rendimiento de sistemas complejos por medio del envío de datos y el control del tiempo de los resultados con un cronómetro. Esta no es la mejor práctica industrial para las pruebas de rendimiento.

La automatización inapropiada usualmente falla. Por ejemplo, observamos a una organización invirtió arriba de US\$ 100.000 en un intento de automatizar las pruebas de instalación. Hablé con otro cliente que perdió US\$ 500.000, en un intento de la automatización de una compleja interfaz de usuario donde el juicio humano era obligatorio para la interpretación de los resultados.

Aquí hay algunas pautas para el éxito con las herramientas de ejecución de pruebas.

Mientras podría parecer que las técnicas de automatización de pruebas de captura y reproducción reducirían los costos de desarrollo, los costos de ejecución y mantenimiento son altos para un gran número de pruebas. Normalmente, no obtendrá un retorno de inversión positivo de esta manera.

En vez de eso, utilice frameworks de pruebas dirigidas por datos que separan las entradas (los datos) de los procedimientos (los guiones) que utilizan los datos. Los expertos en automatización pueden construir el framework y los guiones, mientras que los que no son expertos en automatización pueden crear los datos de prueba y los resultados esperados. Esto funciona.

En algunos casos, las técnicas dirigidas por los datos, en vez de la codificación en duro de las combinaciones de datos en una hoja de cálculo, son capaces de generar datos utilizando algoritmos y parámetros configurables en el tiempo de ejecución. Como ejemplo de esto, considere una herramienta que utiliza un algoritmo para generar identificadores aleatorios de usuarios. Un punto importante aquí es que cuando utilice los valores generados aleatoriamente, asegúrese de utilizar un parámetro configurable en el proceso de generación aleatoria, así que pueda recrear el patrón exacto de las entradas. De otra manera la reproducción de las fallas puede ser difícil.

Alternativamente, utilice frameworks de pruebas dirigidos por palabras clave que utilizan palabras clave (o palabras de acción) en un archivo leído por el guión. Una hoja de cálculo o un archivo plano contiene palabras clave que describen las acciones que deben ser tomadas — de donde proviene la frase “palabras de acción”. El archivo contiene también datos de prueba. De nuevo, los expertos en Automatización pueden construir el framework y los guiones, mientras que los no expertos pueden crear los archivos de palabras clave y definir las palabras clave basadas en la aplicación específica. Por ejemplo, una aplicación bancaria podría tener palabras clave como “adicionar cuenta”, “realizar transferencia” y así sucesivamente.

Tanto para los métodos dirigidos por datos y los dirigidos por palabras clave, la gente que quiere construir el framework necesita experticia técnica en el lenguaje de guión. Es la mejor práctica para separar los expertos del dominio, quienes van a construir las pruebas, de los expertos de automatización quienes construyen las frameworks, porque cada uno posee un conjunto de habilidades única y especial. A menudo es difícil de encontrar a alguien que sea un experto tanto en el dominio que está probando como en la construcción de frameworks de automatización de pruebas.

Como con algunas formas de pruebas, usted necesita un oráculo de pruebas. Una manera de hacerlo así, es de crear una línea de base, la cual es donde los resultados esperados son generados de la aplicación, verificados manualmente (y tal vez corregidos) y guardados para una comparación posterior. Otra opción es la forma automatizada de la generación de los resultados esperados mientras las pruebas se están ejecutando.

Aquí hay algunas pautas para el éxito con otros tipos de herramientas de pruebas:

Cuando se está tratando de crear pruebas de rendimiento o fiabilidad, recuerde que estas pruebas y estas herramientas de pruebas de rendimiento requieren experticia en el rendimiento para diseñar las pruebas e interpretar los resultados.

Cuando se están utilizando herramientas de análisis estático para hacer valer los estándares de código, utilice una introducción en fases para el código existente.

De otra manera, un número gigantesco de falsos positivos ocurrirán, abrumando a su equipo y

posiblemente creando verdaderos problemas de calidad cuando los cambios son realizados para atender las violaciones menores de estándares de código y por consiguiente causando una regresión seria del sistema.

Cuando se está gestionando un trabajo de pruebas complejo, las herramientas de gestión de pruebas pueden realmente ayudar. Sin embargo, la mayoría tienen pocas facilidades para la creación de informes. Sus informes, gráficos y tablas predeterminados tienden a ser extremadamente tácticos y de uso solamente para el Jefe de Pruebas. Entonces, generalmente usted tendrá que exportar los datos a una hoja de cálculo para producir los informes deseados.

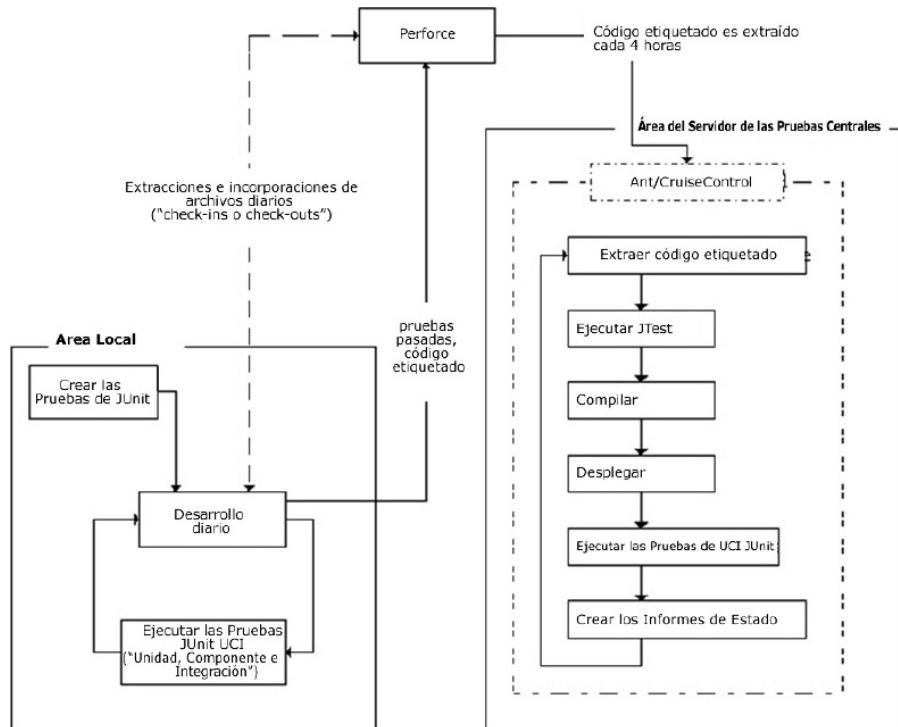


Figura 6.3: Caso de Estudio Estático y de Caja Blanca

Esta figura muestra el caso de estudio de un proyecto en el que nuestros asociados y nosotros trabajamos para un cliente.

Abajo a la izquierda, usted puede observar los programadores creando código Java y pruebas j-Unit en su entorno local de desarrollo y pruebas. Una vez al día, colocan su código y pruebas en el repositorio de fuentes — que resultó ser Perforce32, mostrado en la parte superior — para que estén guardados de forma segura, pero ambos ítems permanecen bajo el control de los programadores.

Una vez que el código está terminado y todas las pruebas unitarias pasan, el programador presenta su código, sus pruebas de unidad y sus resultados de las pruebas de unidad a su líder del desarrollo para la revisión. El revisa estos ítems, y, si son aceptables, los aprueba para una incorporación (“check-in”) oficial al repositorio de código fuente, con una bandera que indica que están terminados. (A propósito, ¿Debería haber incluido este proceso más gente, incluyendo una representación del equipo de pruebas? Si. Sin embargo, eso estaba más allá del nivel de madurez de esta organización en ese momento).

En el entorno de pruebas central, mostrado en el lado derecho de la figura, un guión de pruebas que utiliza Ant y Cruise Control33 comprueba con Perforce cada cuatro horas para ver si algún código nuevo y terminado ha sido incorporado (“checked-in”). Si es así, éste recupera todo el código y todas las unidades de pruebas. Éste utiliza una herramienta de pruebas estáticas, J-Test, para comprobar las violaciones a los estándares del código. Cualquiera de las violaciones es capturada en un informe.

Entonces éste compila una versión de pruebas del código y lo despliega a un entorno de pruebas. Si eso tiene éxito, éste ejecuta cada una de las pruebas de j-unit de unidad, componente e integración contra la versión de pruebas. Todos los resultados son capturados en un informe de pruebas.

Finalmente, basado en los informes de pruebas estáticas y dinámicas, un informe de resumen es creado y publicado en la intranet de compañía. Todos los miembros del equipo reciben un correo electrónico notificándoles dónde ellos pueden encontrar el informe de resumen y cualquier detalle del informe.

El esfuerzo total necesario para construir este framework fue de 20 semanas hombre, a través de nosotros y nuestros dos asociados. Eso incluye el tiempo consumido capacitando a los

programadores en la utilización del framework y el tiempo consumido realizando un análisis de los riesgos para determinar cuáles pruebas deben ser automatizadas. El presupuesto total de la herramienta fue de US\$ 20.000, y eso fue solo para J-Test.

Como puede ver, este método, mientras no es tan glamoroso o ampliamente comercializado como las técnicas de GUI, es económico y rápido. Como una manera de mitigar los riesgos de regresión temprano en el proyecto, es difícil de superar. Para más detalles acerca de como hicimos esto, vea el artículo, "Mission Made Possible — La misión hecha posible"-, en www.rbcus-us.com. Este artículo apareció inicialmente en la revista "Better Software". Agradecemos a nuestro colega y coautor de este artículo, Greg Kubaczowski.

6.2.1 Ejercicios

Ejercicio 1

Basado en la argumentación en esta sección, ¿Cuáles tipos de automatización de pruebas (si algunos) piensa usted que serían exitosos para las pruebas del quiosco?

Basado en la argumentación en esta sección, ¿Cuáles tipos de automatización de pruebas (si algunos) piensa usted que serían exitosos para las pruebas del centro de datos?

¿Sugeriría la automatización de las pruebas de punta a punta (del quiosco al centro de llamadas o del centro de llamadas al quiosco), y, si es así, cómo?

Argumente.

Solución del Ejercicio 1

Ciertamente consideraríamos pruebas aleatorias dirigidas por palabras clave para los quioscos. Estos tipos de herramientas de pruebas son a veces llamados "monos— en inglés: monkeys". Una búsqueda en la Web de las frases "monos tontos probando — en inglés: testing dumb monkeys" y "pruebas aleatorias" brindaran información interesante. Ejecutariamos estas pruebas a través de la interfaz gráfica del usuario, que serían los navegadores compatibles con el quiosco. Porque el quiosco es compatible con tres navegadores, la herramienta elegida debe ser compatible con ellos también. Utilizaríamos una herramienta de pruebas de interfaz de usuario gráfica — GUI, para realizar las pruebas de cobertura de las funciones dirigidas por palabras clave para el centro de llamadas.

Por supuesto hay herramientas comerciales para esto, así como Segue y las ofertas de Mercury. También existen opciones de software libre.³⁴

A largo plazo, quisiéramos automatizar las pruebas de punta a punta por medio de la extensión de las herramientas de interfaz gráfica de usuario — GUI — en el quiosco y el centro de llamadas para coordinar con cada uno. Nuestros asociados y nosotros hicimos esto para un cliente en un proyecto, donde teníamos un servidor de respuesta de voz interactiva, dirigido por una herramienta casera de pruebas de telefonía y un centro de llamadas dirigido por el producto QA Partner³⁵ de Segue (luego de SilkTest) con un ejecutivo conectándolos a ambos, como se ve en la figura 6.4.

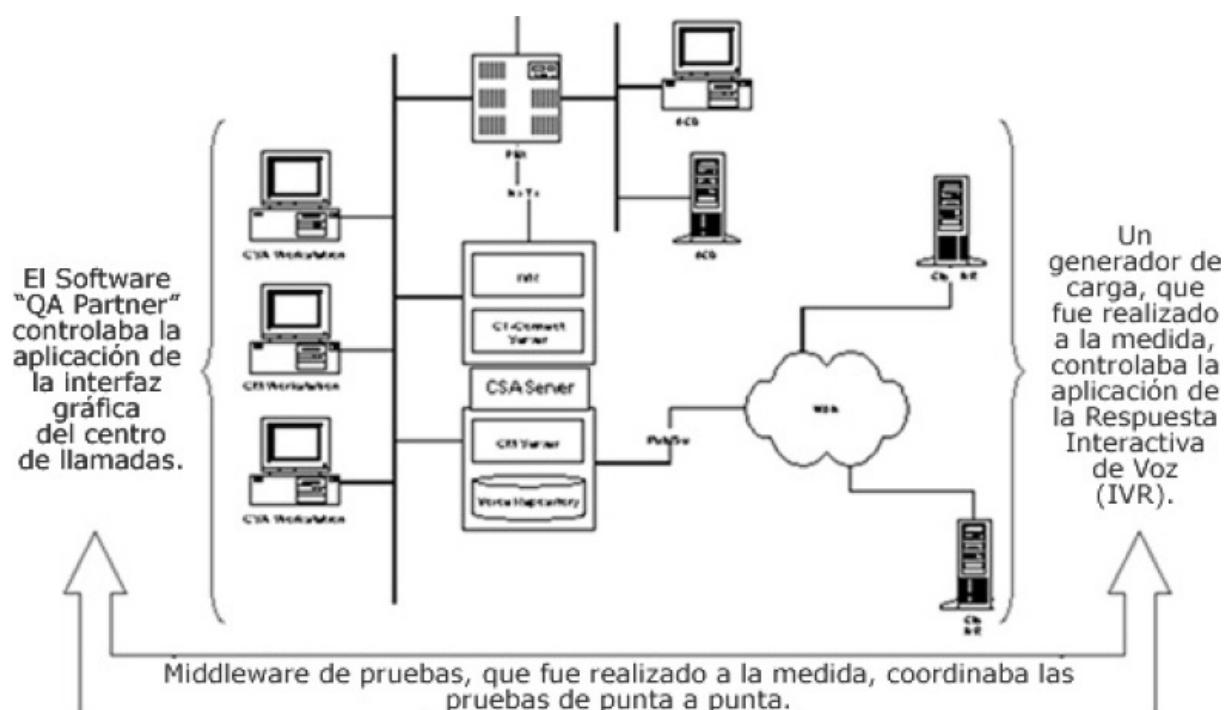


Figura 6.4: Una arquitectura del sistema de pruebas automatizadas coordinando las pruebas de punta a punta.

6.3 Introducción de una herramienta en una organización

Objetivos del Aprendizaje

- LO-6.3.1 Formular los principales principios de la introducción de una herramienta en una organización. (K1)
- LO-6.3.2 Formular las metas de una prueba de concepto para la evaluación de herramientas y de una fase piloto para la implementación de herramientas. (K1)
- LO-6.3.3 Reconocer esos otros factores necesarios para el buen soporte de herramientas más que simplemente la adquisición de una herramienta. (K1)

Esta sección, Introducción de una herramienta en una organización, cubrirá los siguientes conceptos clave.

- Introducción de una herramienta en una organización.
- Metas de una prueba de concepto o proyecto piloto para la evaluación de una herramienta.
- Factores necesarios para un buen soporte de una herramienta.

Cualquier decisión para la automatización de las pruebas es una gran decisión. Ésta cambiará la manera en la cual su proceso de pruebas funciona. En el caso de éxito, la automatización de las pruebas le permitirá alcanzar objetivos actuales—y tal vez nuevos—del equipo de pruebas en una manera más efectiva y eficiente. La selección de la herramienta correcta es una gran parte de la automatización exitosa.

Entonces, ¿Cuáles factores debería considerar cuando seleccione una herramienta de pruebas?

Primero, debería realizar una evaluación realista de la madurez del proceso de pruebas, el proceso de desarrollo de software en general y la organización como un todo. ¿Cuáles son las fortalezas y debilidades? ¿Cuáles son las oportunidades para el proceso de pruebas, el proceso de desarrollo y la mejora organizacional creados por la utilización de las herramientas de pruebas automatizadas? ¿Puede usted y su organización explotar completamente esas oportunidades? Usted debería tener respuestas claras y realistas para esas preguntas.

Basado en su evaluación, debería definir requisitos claros y criterios objetivos para la herramienta. Puede utilizar estos requisitos y criterios para evaluar las herramientas disponibles. Recuerde de considerar no solo las herramientas comerciales, pero también el software libre, de código abierto y herramientas de shareware. Utilizando estos requisitos y criterios, debería ser capaz de eliminar la mayoría de las herramientas que no funcionarán para usted y luego puede concentrarse en una lista corta de posibles herramientas.

En este punto, es buena idea de llevar a cabo una prueba de concepto de una o más herramientas. Esto involucra la utilización de las herramientas candidatas de pruebas para determinar si ellas pueden probar el sistema sometido a pruebas efectivamente y eficientemente en la infraestructura actual. Si no es posible, ¿Hay cambios factibles y prácticos para aquella infraestructura que resolvieran los problemas? ¿Puede lograr todos sus objetivos para la automatización con esta herramienta, o tendrá que comprometerse con algunas de éstas? Si es así entonces, ¿Son aquellos compromisos aceptables? Idealmente usted realizaría esta prueba de concepto para todas las herramientas de su lista corta, aunque muchas organizaciones lo encuentran demasiado laborioso. Por lo menos realice esta prueba de concepto con la mejor herramienta de su lista corta antes de comprometerse con esa herramienta. Muchos equipos han aprendido la sabiduría de un viejo cliché “Cásate demasiado pronto y te arrepentirás demasiado tarde, después de un casamiento a la fuerza con la herramienta de pruebas”

También debería evaluar al vendedor. Si está considerando herramientas comerciales, analice muy de cerca la capacitación y el soporte, así como los aspectos comerciales como la estabilidad de la compañía. Para herramientas de código abierto y herramientas libres, considere la fortaleza y estabilidad de la comunidad que apoya el proyecto.

Como un área de consideración, debería evaluar sus necesidades de capacitación. Asegúrese de tomar en cuenta las habilidades vigentes acerca de la automatización del equipo de pruebas. Sólo porque el proveedor de la herramienta tiene un curso de capacitación, no significa que es factible — o esté dentro del presupuesto — que cada uno pueda asistir a ese curso.

Asegúrese de identificar sus requisitos internos para la orientación y tutoría. ¿Cómo los realizaría? Alguien tendrá que ayudar a difundir la práctica de la automatización de pruebas en toda su organización. Usted quisiera que la creación, la ejecución y el mantenimiento de los productos del trabajo de automatización sean realizados consistentemente. Sin orientación y tutoría—junto con los estándares para la utilización de la herramienta—las organizaciones ven mucha divergencia en las prácticas, fallas locales de automatización en algunos proyectos y oportunidades perdidas para la reutilización, cuando los distintos componentes desarrollados por diferentes probadores no funcionan juntos.

Finalmente, comprenda el caso del negocio, que lo hemos tratado en la sección previa. Asegúrese de saber cómo establecerá el punto de equilibrio (“breakeven”) — el momento cuando los

beneficios acumulados hayan alcanzado el costo de la inversión — y cómo medirá y creará informes acerca del retorno de inversión en el tiempo. Recuerde que dos declaraciones de gestión claves aplican: Lo que es medido es realizado y lo que es validado es financiado. Si no puede medir el valor de la automatización de pruebas, pronostique ¿Cuál es un candidato para el desfinanciamiento cuando el dinero no alcanza?

Porque la selección de la herramienta correcta—y evitando los errores de las herramientas clásicas —es como una parte crítica de la introducción exitosa de una herramienta de pruebas en una organización, nosotros repasaremos este tema nuevamente al final de esta sección.

La automatización exitosa requiere la eliminación de los obstáculos para el éxito—antes que comience.

Un obstáculo para el éxito que a menudo encontramos con los clientes que tratan de automatizar son los procesos de pruebas caóticos, reactivos y cortos de tiempo. Si cada día es un nuevo día en su grupo de pruebas, y no puede predecir en lo que trabajará el próximo día cuando se va a casa en la noche, no hay absolutamente ninguna razón para tratar de automatizar. Recuerde, el ROI de la automatización viene de la repetición de las pruebas una y otra vez. Entonces, primeramente tiene que tener el proceso de pruebas manual en control y luego automatizar.

Otro obstáculo es el personal con habilidades insuficientes. La automatización de pruebas es casi la tarea más dura de programación que hay, a la altura de la programación de sistemas operativos. Entonces, si quiere que su proyecto de automatización falle, una gran manera de asegurar eso, es de utilizar la capacitación durante el trabajo de los ingenieros de pruebas no calificados, poniéndolos a crear las pruebas automatizadas. En vez de perder tiempo y dinero de esa manera, debería contratar gente que tenga **por lo menos** cinco años de experiencia en la automatización de pruebas—entre éxitos y fracasos—y también capacitar su personal que trabajará en esto y utilizará las pruebas automatizadas.

Otro obstáculo para el éxito es que un sistema sea inestable y cambie rápidamente. Esto nos conduce a costos enormes e impredecibles del mantenimiento de las pruebas que arruinan el ROI y la utilidad práctica de sus pruebas automatizadas. Entonces, primeramente estabilice el sistema — especialmente en las interfaces probadas como las de GUI y API.

Ahora, el mayor obstáculo que encontramos con los clientes una y otra vez son las expectativas poco realistas de la dirección de lo que la automatización producirá. Estas expectativas nacen de dos factores principales. Uno es la frustración de la dirección con el costo y la duración de la ejecución de las pruebas—a menudo erróneamente atribuido a los probadores en vez de al gran número de defectos puestos en los objetos de prueba durante los requisitos, el diseño y la implementación. El otro es el bombardeo de una exagerada promoción de ventas y marketing en la conciencia de la ingeniería del software por los vendedores de herramientas. Entonces, debe comunicar efectivamente los beneficios, las limitaciones, el caso del negocio (incluyendo un modelo sólido del retorno de la inversión) y los costos de la automatización. No comience a automatizar hasta que tenga un financiamiento del negocio basado en expectativas realistas.

Finalmente, otro obstáculo es la falta de una apropiada herramienta o un oráculo práctico para atender los riesgos de calidad importantes. En este caso, puede elegir la construcción de su propia herramienta u oráculo—o esperar a que el mercado produzca algo mejor. Ya que la construcción de herramientas es muy cara y riesgosa, es usualmente mejor esperar al menos que la necesidad sea urgente.

Las organizaciones ignoran esta lista de obstáculos bajo su seria responsabilidad. Cuando estos problemas permanecen, cualquier intento de automatización normalmente falla. Esto no es solamente una gran pérdida de tiempo y dinero, sino que el fracaso crea una resistencia organizacional para los proyectos subsiguientes de automatización. Si aprende la lección de tomar en serio estos obstáculos, que hacen fracasar, es posible que no tenga la posibilidad de fracasar otra vez.

Asumamos que hemos eliminado los obstáculos y queremos comenzar con la automatización. Como decimos a menudo, debe comerse el sándwich poco a poco. La mejor manera de comenzar la automatización es con un proyecto piloto.

Sus metas en este proyecto piloto deberían ser las siguientes:

Aprender más acerca de la herramienta y cómo utilizarla.

Adaptar la herramienta, los procesos y las prácticas para que se ajusten a su organización y sus sistemas.

Estandarizar las formas de utilización, gestión, guardado y mantenimiento de la herramienta y de los activos de las pruebas que la herramienta crea.

Evaluuar el retorno de la inversión.

Cometerá algunos errores en el proyecto piloto, entonces escoja un proyecto que pase desapercibido y es de bajo riesgo. De esa manera los errores no se verán exagerados por una publicidad organizacional deficiente acerca de sus problemas.

¿Cuáles son algunos factores de éxito para el despliegue de herramientas de pruebas?

Como dijimos antes, cómase el sándwich paso a paso. Despliegue la herramienta incrementalmente en el resto de la organización, construyendo un proyecto de automatización exitoso. Si tiene un proyecto de automatización no exitoso en el camino, asegúrese de incorporar las lecciones aprendidas antes de continuar.

La utilización de las herramientas de automatización de pruebas cambiará—por lo menos debería o debe—la manera que usted controla el proceso de pruebas, completamente desde la planificación hasta el cierre. Asegúrese de adaptar y mejorar los procesos de pruebas para ajustar la utilización de las herramientas.

Mientras la automatización de pruebas puede hacer la vida de los probadores más fácil, las pruebas automatizadas son complejas y a menudo bestias sensibles que proporcionan resultados erróneos e inválidos si son utilizadas incorrectamente. Por lo tanto, proporcione capacitación, entrenamiento y tutorías para los nuevos usuarios.

Para aquellos que utilizan las herramientas y especialmente aquellos que construyen los casos de prueba y los frameworks de prueba, defina guías de uso de la herramienta. La automatización de pruebas es desarrollo de software y el desarrollo de software sin consistencia y diseño es un desastre.

Aprenda las maneras de mejorar continuamente la utilización de las herramientas. Las herramientas y las técnicas están siempre evolucionando.

Proporcione el soporte para el equipo de pruebas de las herramientas que utiliza. Como mencionamos anteriormente cuando abordamos la selección de las herramientas, es donde su plan entra en acción para la orientación y tutoría.

Sin embargo, el flujo de la información no es sólo hacia el exterior de los tutores a los equipos que utilizan las herramientas. Debería recopilar las lecciones aprendidas de todos los equipos que emplean las herramientas. Algunas veces a las organizaciones les gusta tener un “comité de automatización” u otro equipo de toda la organización que se reúne para compartir las mejores prácticas, historias de éxito y por supuesto los cuentos con moraleja.

Finalmente, como lo mencionamos anteriormente, asegúrese de monitorear la utilización y los beneficios de la herramienta. No sólo debería tener un caso de negocios y un modelo ROI, debería utilizarlo regularmente para medir e informar el ROI. Comprenda el ROI que está obteniendo y encárguese del problema si no está obteniendo el ROI que esperó.

Aquí hay una lista final de las trampas de las herramientas para mantenerlo alerta a medida que avance, extraída del libro de Rick Craig, *Systematic Software Testing*.

Una trampa común es la falta de una estrategia clara. Las organizaciones solo compran las herramientas y después esperan que algo bueno ocurra. Esto no ocurre usualmente.

Como mencionamos antes, las expectativas poco realistas son un gran problema. Encárguese de estos problemas donde existan.

Otra trampa es la falta de participación de los interesados del negocio. Esto no se refiere sólo a los probadores, sino que a otros en los proyectos quienes serán afectados por las pruebas automatizadas. El riesgo principal es la falta de participación durante el trabajo de la automatización.

Algunas organizaciones tratan de utilizar la capacitación de las herramientas durante el trabajo, algunas veces por que no pueden encontrar o pagar por una capacitación adecuada y de alta calidad. Eso resulta en un fracaso de la automatización de las pruebas.

Algunas organizaciones tratan de automatizar las cosas equivocadas, las pruebas equivocadas, como mencionamos en una sección anterior.

Algunas organizaciones eligen la herramienta equivocada o el proveedor equivocado de las herramientas. Recuerde, la fuerza del esfuerzo de las ventas y del marketing asociado con una herramienta de pruebas le dice poco o nada acerca de su utilidad. Sólo un cuidadoso proceso de selección de las herramientas puede resultar en la selección correcta de la herramienta. Aquí **no busque atajos**, o pagará el precio después.

Cuando evalúe una herramienta, sea realista acerca de las habilidades de la gente involucrada. No compre una herramienta que tiene problemas de usabilidad si tiene probadores no técnicos quienes crearán casos de prueba y resultados esperados. La gente frustrada comete errores, la gente frustrada no es productiva, y la gente frustrada habla mal de la fuente de su frustración. ¿Por qué hacer la automatización más difícil de lo que ya inherentemente es?

Como hemos mencionado antes, necesita tener un sistema estable para la exitosa automatización de pruebas, pero mucha gente trata de automatizar un sistema inestable sometido a pruebas. Esto usualmente fracasa.

Toma tiempo el despliegue exitoso de la automatización de pruebas en una organización. Si trata de hacer demasiado, demasiado pronto, tendrá que lidiar con un fracaso grande, notable y costoso —o con algunos de esos. La siguiente noticia que usted escuche será que la dirección está cancelando la automatización de pruebas, a menudo para bien.

Algunas organizaciones subestiman el tiempo y los recursos necesarios para la automatización, como hemos mencionado antes. Tenga una estimación realista para el esfuerzo, junto con un caso de negocios sólido.

Algunas organizaciones tratan de automatizar sin invertir en entornos de pruebas automatizados adecuados, únicos y dedicados. Éstas tratan de ejecutar las pruebas de rendimiento en entornos diferentes al de producción. Éstas tratan de ejecutar las pruebas manuales y las pruebas automatizadas en el mismo entorno al mismo tiempo. Estos esfuerzos resultan en resultados de pruebas equivocados, tiempo perdido, y así sucesivamente.

Algunas organizaciones toman el método correcto, pero en el momento equivocado. Si la organización está en medio de una crisis existencial relacionada con algún proyecto crítico, ése es el proyecto equivocado para ser utilizado como un proyecto piloto para la automatización, a menos que no haya otra manera de probar lo suficiente en ese proyecto.

Finalmente, hay la trampa de quedarse sin dinero antes de que los beneficios hayan comenzado a acumularse. Esto es a menudo—pero no siempre—asociado con la subestimación del tiempo y los recursos necesarios. Sin embargo, en algunos casos la dirección inconsiguiente retira la aprobación del proyecto de automatización basada en el cambio de prioridades. Es triste cuando esto ocurre, pero ciertamente ocurre.

Hemos hablado mucho de los puntos negativos asociados con la automatización de pruebas, pero no porque no creamos en ésta. Creemos en ésta. RBCS/Business Innovations han realizado decenas de proyectos de automatización de pruebas para clientes y estamos orgullosos de realizarlos.

Sin embargo, nos gusta hacer proyectos que tengan éxito y no proyectos que fracasan. Entonces, para asegurar que usted tenga éxito, construya un caso de negocios de la automatización de pruebas, rompa los obstáculos, evite las trampas de las herramientas de pruebas, realice un proyecto piloto y desplíquelo incrementalmente. ¡Buena suerte!

Glosario del ISTQB

Herramienta de revisión: Una herramienta que proporciona apoyo al proceso de revisión. Las características típicas incluyen la planificación de la revisión y el apoyo del seguimiento, el apoyo de las comunicaciones, las revisiones colaborativas y un repositorio para recopilar las métricas e informar acerca de éstas.

Herramienta de pruebas de estrés: Una herramienta que apoya las pruebas de estrés.

Pruebas de estrés: Un tipo de pruebas de rendimiento conducido para evaluar un sistema o componente en o más allá de los límites de sus cargas de trabajo anticipadas o especificadas o con una reducida disponibilidad de recursos así como el acceso a la memoria o a los servidores. Véase también las pruebas de rendimiento y pruebas de carga. Note que este término no es específicamente enunciado en esta sección, pero está incluido aquí para ayudarle a comprender el término herramienta de pruebas de estrés.

Herramienta de preparación de datos de prueba: Un tipo de herramienta de pruebas que hace posible la selección de los datos de bases de datos existentes o que son creados, generados, manipulados y editados para su utilización en las pruebas.

Herramienta de ejecución de pruebas: Un tipo de herramienta de pruebas que es capaz de ejecutar otro software utilizando un guión de pruebas automatizadas, p.ej. la captura o la reproducción.

Herramienta de gestión de pruebas: Una herramienta que proporciona el soporte para la gestión de pruebas y el control de la parte de un proceso de pruebas. A menudo tiene varias capacidades, así como la gestión del testware, la creación de cronogramas de las pruebas, el registro de los resultados, el seguimiento del progreso, la gestión de incidencias y los informes de pruebas.

Herramienta de frameworks de pruebas unitarias: Este término no está definido en el glosario del ISQTB. Sin embargo está relacionado con el término framework de pruebas unitarias.

Framework de pruebas unitarias: Una herramienta que proporciona un entorno para las pruebas de unidad o componente en la cual un componente puede ser probado aisladamente o con stubs y drivers adecuados. También proporciona otro apoyo para el desarrollador, así como las capacidades de depuración.

6.3.1 Ejercicios

Ejercicio 1

¿Utilizaría el centro de llamadas o el quiosco como un piloto para la automatización?

¿Cuáles barreras debería superar para realizar un piloto con la herramienta contra el centro de llamadas o el quiosco?

¿Cuáles trampas de pruebas debería evitar?

Argumente.

Solución del Ejercicio 1

Por un lado, el centro de llamadas es un piloto de automatización atractivo por que puede utilizar herramientas comerciales. Además, mientras que el riesgo de negocios es alto, el riesgo técnico es relativamente bajo para el centro de llamadas, lo que significa que las pruebas probablemente la mayoría de las veces pasarián y así son más eficientes para automatizar. Ésta es la más costosa pero la menos riesgosa de las dos.

Por el otro lado, el quiosco es un piloto de automatización atractivo porque usted puede utilizar

herramientas libres, de código abierto, con cierto grado de personalización. El riesgo técnico y de negocios son ambos altos, entonces la rentabilidad para la automatización es clara y las pruebas revelarán defectos. Es la opción menos costosa, ciertamente en cuanto a los costos fijos, pero esto es también más riesgoso por que las herramientas de código abierto tienen menos antecedentes.

Personalmente, nosotros comenzaríamos con el quiosco. Una vez que tuviéramos las pruebas automatizadas ejecutándose allí y hubiéramos demostrado la rentabilidad de la automatización a través de los defectos encontrados y las ejecuciones rápidas de pruebas de regresión, utilizaríamos esa historia de éxito para venderle un buen argumento a la dirección acerca de las costosas licencias para las pruebas del centro de llamadas.

La automatización de los quioscos enfrentaría cuatro barreras principales. Las primeras dos son cuestiones humanas. La barrera más importante — y de hecho omnipresente — es la de las expectativas poco realistas de la dirección. Nos aseguraríamos que tuviéramos un caso de negocios fuerte que le hubiéramos vendido a la dirección acerca de lo que la automatización de las pruebas de los quioscos podría o no podría hacer. La otra barrera humana de las habilidades emerge de las pruebas del quiosco con herramientas de código abierto. Tendríamos que lidiar con el aumento de nuestro equipo. Probablemente contratariamos a un contratista o consultor como el ingeniero de pruebas del quiosco quién ya tuviera la experticia y haríamos transferir ese conocimiento a otros como parte de su descripción del puesto de trabajo.

Las segundas dos barreras son cuestiones técnicas. Primero, es probable que Omninet esté cambiando rápidamente en la interfaz del usuario. Entonces, quisiéramos realizar las pruebas muy flexibles para esos cambios, aún en el riesgo de incrementar la probabilidad de que se escapen defectos. Después de todo, estamos planificando realizar pruebas exploratorias para asegurar que no se nos escape nada, así las pruebas automatizadas no tienen que capturar cada tipo de defectos funcionales. Este método nos conduciría a resolver también la cuarta barrera, la falta de un oráculo de pruebas completo. Dejaremos de lado esta barrera utilizando un oráculo parcial.

Las trampas de las herramientas que hay que evitar incluyen en primer lugar la selección de la herramienta equivocada. Especialmente con herramientas de código abierto, existe la posibilidad de que un número de usuarios cada vez más escaso dejará a la herramienta huérfana. Usted quisiera evaluar este riesgo, y asegurarse de que se siente cómodo manteniendo la herramienta usted mismo si eso ocurre. Cuidadosamente evalúe la funcionalidad, la usabilidad y la mantenibilidad de varias herramientas y reconsideré la automatización completamente, si no puede encontrar una herramienta que va a satisfacer las necesidades a largo y a corto plazo.

También tiene que evitar cometer la equivocación común de subestimar cuánto tiempo tomará y qué tan difícil será introducir la herramienta y automatizar las pruebas. Esta equivocación creará expectativas equivocadas a la dirección, seguida de expectativas no alcanzadas y típicamente seguida por la cancelación del proyecto de automatización y frecuentemente despidos. Los jefes de pruebas y los probadores inteligentes son cuidadosos para permitir mucho tiempo para la automatización.

Otro método que ayuda es el incremental, construyendo e introduciendo pequeños pedazos de automatización en forma continua en vez de un gran paquete.

Preguntas de Examen de Muestra y Simulación

Para finalizar cada capítulo, usted puede tratar de resolver una o más preguntas de examen de muestra para reforzar su conocimiento y comprensión del material y prepararse para el examen del Probador ISTQB Nivel Básico.

Sección 6.1 Tipos de herramientas de pruebas (K2)

Términos

Herramienta de gestión de configuración, herramienta de cobertura, herramienta de depuración, herramienta de análisis dinámico, herramienta de gestión de incidencias, herramienta de pruebas de carga, herramienta de modelado, herramienta de monitoreo, herramienta de pruebas de rendimiento, efecto sonda, herramienta de gestión de requisitos, herramienta de revisión, herramienta de seguridad, herramienta de análisis estático, herramienta de pruebas de estrés, comparador de pruebas, herramienta de preparación de datos de prueba, herramienta de diseño de pruebas, arnés de pruebas, herramienta de ejecución de pruebas, herramienta de gestión de pruebas y herramienta de frameworks de pruebas unitarias.

#	Pregunta	K
191.	Objetivo del aprendizaje: LO-6.1.1 Usted está trabajando como un jefe de pruebas en un proyecto grande. Usted necesita una herramienta que gestionará la trazabilidad de las pruebas, los resultados de las pruebas y los defectos a las bases de las pruebas. ¿Qué tipo de herramienta necesita usted?	2

	A. Una herramienta de análisis estático. B. Una herramienta de gestión de pruebas. C. Una herramienta de seguridad. D. Una herramienta de medición de la cobertura.	
192.	Objetivo del aprendizaje: LO-6.1.2 ¿Cuál de las siguientes herramientas de pruebas son probables que sean de mayor interés para los desarrolladores? A. Un arnés de pruebas unitarias. B. Una herramienta de gestión de pruebas. C. Una hoja de cálculo. D. Una herramienta de seguimiento de defectos.	1
193.	Objetivo del Aprendizaje: LO-6.1.3 ¿Cuál de los siguientes es el mejor ejemplo de la utilización de una herramienta de prueba para apoyar a las pruebas? A. La realización de una revisión guiada. B. El informe de un defecto. C. La utilización de un comparador. D. La verificación de la trazabilidad.	2
194.	Objetivo del aprendizaje: término. 1 ¿Qué es el efecto sonda? A. Las pruebas con el objetivo de mostrar que un componente o sistema no funciona. B. Las condiciones del entorno y el estado que deben ser cumplidas después de la ejecución de una prueba o un procedimiento de prueba. C. Algo que pasa con el software satélite. D. El efecto en el componente o sistema cuando está siendo medido.	1

Sección 6.2: Utilización efectiva de las herramientas: beneficios y riesgos potenciales (K2)

Término

(Pruebas) Dirigidas por datos, (Pruebas) Dirigidas por palabras clave y lenguaje de guión.

#	Pregunta	K
195.	Objetivo del aprendizaje: LO-6.2.1 Usted está trabajando en un proyecto que está realizando una versión de mantenimiento de un sistema bancario grande y de alto riesgo. La gente está muy preocupada por la regresión de la funcionalidad existente, y desea ejecutar nuevamente el conjunto existente de 12.254 casos de prueba contra cada versión de pruebas. ¿Cuál de los siguientes es un beneficio de la automatización de las pruebas para este proyecto y cuál es probable que interese al equipo del proyecto? A. La reducción del trabajo repetitivo. B. La facilidad de archivar los informes de defectos. C. El bajo costo de la automatización de todas las pruebas. D. La oportunidad de aprender una nueva habilidad.	2
196.	Objetivo del aprendizaje: LO-6.2.2 Considere lo siguiente: I. Dirigidas por datos. II. Exploratorias. III. Dirigidas por palabras clave. IV. Portabilidad. V. CMMI. ¿Cuál de las siguientes afirmaciones es verdadera acerca de las técnicas de guiones para la ejecución de pruebas automatizadas? A. I y III son técnicas reconocidas para las pruebas automatizadas efectivas y eficientes. B. I, II, III, IV y V son técnicas reconocidas para las pruebas automatizadas efectivas y eficientes. C. II y IV son técnicas reconocidas para las pruebas automatizadas efectivas y eficientes. D. Ninguna de ellas son técnicas reconocidas para las pruebas automatizadas efectivas y eficientes.	1

197.	<p>Objetivo del aprendizaje: término ¿Qué es un lenguaje de guión?</p> <ul style="list-style-type: none"> A. El proceso de pruebas para determinar la portabilidad de un producto de software. B. Un dispositivo, programa o sistema de informático utilizado durante las pruebas, el cual se comporta u opera como un sistema determinado. C. Un documento que especifica una secuencia de acciones para la ejecución de una prueba. D. Un lenguaje de programación en el cual son escritos los guiones de pruebas. 	1
------	--	---

Sección 6.3: Introducción de una herramienta en una organización (K1)

Términos

No hay términos específicos.

#	Pregunta	K
198.	<p>Objetivo del aprendizaje: LO-6.3.1 ¿Cuál de los siguientes es un principio importante que ayudará en la introducción de una herramienta de pruebas?</p> <ul style="list-style-type: none"> A. Evaluar al proveedor. B. Confiar en el proveedor. C. Automatizar todas las pruebas inmediatamente. D. Eliminar las pruebas que no son automatizables. 	1
199.	<p>Objetivo del aprendizaje: LO-6.3.2 ¿Cuál de las siguientes es una meta inteligente para un piloto de automatización de pruebas?</p> <ul style="list-style-type: none"> A. Completar la automatización de todas las pruebas. B. Evaluar la situación del costo/beneficio para la automatización. C. Determinar la probabilidad y el impacto de los riesgos del producto. D. Emplear contratistas de bajo costo, a corto plazo para todas las tareas de automatización. 	1
200.	<p>Objetivo del aprendizaje: LO-6.3.3 ¿Cuál de los siguientes es un factor no relacionado a la selección de herramientas que influencia en el éxito del despliegue de una herramienta?</p> <ul style="list-style-type: none"> A. El monitoreo de la utilización y los beneficios de la herramienta. B. La evaluación contra requisitos claros y criterios objetivos. C. La evaluación del proveedor, incluyendo la capacitación y el soporte. D. La identificación de los requisitos internos para la tutoría. 	1

Capítulo 6 Pregunta de todas las secciones

#	Pregunta	K
201.	<p>Cubre: las secciones 1 y 3</p> <p>Los clientes se quejan de que el sitio Web de su empresa es demasiado lento. Se le ha pedido que realice la gestión de la parte de las pruebas de un proyecto, que se encarga de reducir la incidencia de esos problemas para la parte del comercio electrónico (tienda en línea) del sitio web de su empresa. Por lo tanto, su trabajo en este proyecto es probable que incluya ¿qué?</p> <ul style="list-style-type: none"> A. Una introducción completa de una herramienta de pruebas de regresión. B. Un proyecto piloto con una herramienta de análisis dinámico. C. La introducción de una nueva herramienta de gestión de configuración en toda la empresa. D. Un proyecto piloto con una herramienta de pruebas de rendimiento. 	2

Preguntas del Examen de Simulación 1

#	Pregunta	K
202.	<p>Objetivo del aprendizaje: término.</p> <p>¿Qué es un stub?</p> <ul style="list-style-type: none"> A. El efecto en un componente o sistema cuando está siendo medido. B. Una implementación esquemática o con propósito especial de un componente de software, utilizada para desarrollar o probar un componente que invoca, o es dependiente de ésta. C. Un entorno de pruebas necesario para conducir una prueba. D. Un componente de software o herramienta de pruebas que reemplaza a un componente, que se encarga del control y/o la invocación de un componente o sistema. 	1
203.	<p>Objetivo del aprendizaje: LO-6.1.1</p> <p>Actualmente, usted está comprometido en trabajar como un jefe de pruebas en un proyecto grande. Usted anticipa un gran número de defectos en el sistema. ¿Cuál clase de apoyo de herramientas podría ser útil para usted como un jefe de pruebas?</p> <ul style="list-style-type: none"> A. Gestión de incidencias. B. Análisis estático. C. Ejecución de pruebas. D. Comparador de pruebas. 	2
204.	<p>Objetivo del aprendizaje: LO-6.2.2</p> <p>¿Qué son los guiones de pruebas automatizadas dirigidas por datos?</p> <ul style="list-style-type: none"> A. Una técnica de guión que utiliza archivos de datos para contener no solamente datos de prueba y resultados esperados, sino también palabras clave relacionadas a la aplicación que está siendo probada. B. Una técnica de guión que guarda las entradas de prueba y resultados esperados en una tabla u hoja de cálculo, de manera que un sólo guión de control pueda ejecutar todas las pruebas en la tabla. C. Una declaración de los objetivos de las pruebas, y posiblemente las ideas de las pruebas acerca de cómo probar. D. Datos que existen (por ejemplo, en una base de datos) antes de que una prueba sea ejecutada, y que afectan o que son afectados por el componente o sistema sometido a pruebas. 	1
205.	<p>Objetivo del aprendizaje: LO-6.3.3</p> <p>Considere los siguientes factores:</p> <ol style="list-style-type: none"> I. Introducción de la herramienta gradualmente. II. Adaptación y mejora de los procesos para adecuarse a la herramienta. III. Suministro de capacitación para los nuevos usuarios. IV. Definición de guías de usuario. V. Implementación para mejorar continuamente la utilización de las herramientas. <p>¿Cuál de las siguientes afirmaciones es verdadera?</p> <ul style="list-style-type: none"> A. Todos estos factores son importantes para el exitoso despliegue de una herramienta de pruebas. B. Los factores I, II y III son importantes para el exitoso despliegue de una herramienta de pruebas, pero el IV y V no son importantes. C. Los factores I, III, IV y V son importantes para el exitoso despliegue de una herramienta de pruebas, pero el II no es importante. D. El factor I es importante para un exitoso despliegue de una herramienta de pruebas, pero el II, III y V no son importantes. 	1

Preguntas del Examen de Simulación 2

#	Pregunta	K
206.	<p>Objetivo del aprendizaje: LO-6.1.1</p> <p>¿Cuál de las siguientes herramientas es la más útil para el análisis cuantitativo (métricas) de pruebas?</p> <ul style="list-style-type: none"> A. Herramienta de gestión de pruebas. B. Herramienta de pruebas estáticas. C. Comparador de pruebas. D. Herramienta de seguridad. 	1

207.	<p>Objetivo del aprendizaje: LO-6.2.1 2</p> <p>Usted acaba de finalizar un proyecto piloto de automatización de pruebas. Fue capaz de automatizar un gran número de pruebas de regresión y lograr una amplia cobertura de las bases de pruebas. Excedió sus propias expectativas acerca de lo que podía ser automatizado en el piloto. Sin embargo, el patrocinador del proyecto no está feliz con el resultado, y cree que todas las pruebas deberían haber sido automatizadas en el piloto.</p> <p>¿Cuál de los siguientes podría ser un riesgo, el cual debería haber sido gestionado más cuidadosamente para este piloto de automatización?</p> <ul style="list-style-type: none"> A. La subestimación del tiempo, costo y esfuerzo para la introducción inicial de una herramienta. B. La subestimación del esfuerzo necesario para mantener los activos generados de las pruebas por la herramienta. C. La dependencia excesiva de la herramienta. D. Expectativas no realistas para la herramienta 	2
208.	<p>Objetivo del aprendizaje: LO-6.3.2</p> <p>Usted acaba de completar un proyecto piloto para una herramienta de pruebas de regresión. Comprende la herramienta mucho mejor, y ha adaptado su proceso a esta. Ha estandarizado un método para utilizar la herramienta. ¿Cuál de las siguientes es una meta de un típico proyecto piloto de automatización de pruebas que queda por ser realizada?</p> <ul style="list-style-type: none"> A. Aprender más detalles acerca de la herramienta. B. Observar cómo la herramienta encajaría en los procesos y prácticas existentes, y cómo estos necesitarían cambiar. C. Decidir acerca de las formas estándares de utilizar, gestionar, guardar y mantener la herramienta y los activos de las pruebas. D. Evaluar si los beneficios serán alcanzados en un costo razonable. 	1
209.	<p>Objetivo del aprendizaje: término</p> <p>¿Qué es una herramienta de análisis estático?</p> <ul style="list-style-type: none"> A. Una herramienta que analiza los artefactos del software, p.ej. los requisitos o el código, sin la ejecución de estos artefactos del software. B. Una herramienta que evalúa el comportamiento, p.ej. el rendimiento de la memoria, el uso del CPU de un sistema o componente durante la ejecución. C. Una herramienta que compara los resultados reales con los esperados, utilizada mientras el software está siendo ejecutado, por ejemplo por una herramienta de ejecución de pruebas. D. Un dispositivo, un programa de computadora, o sistema que acepta las mismas entradas y produce los mismos resultados como un sistema determinado. 	1

31 Véase el libro, *Managing the Testing Process* de Rex Black, 2e, acerca de varios ejemplos de la construcción de herramientas de pruebas de grado profesional utilizando aplicaciones usuales de escritorio.

32 Es un programa cliente/servidor de la empresa Perforce Software que gestiona una base de datos central y un repositorio maestro de las versiones de los archivos.

33 CruiseControl es una aplicación de código abierto basado en Java que permite la compilación automática de proyectos Java, utilizando Ant o Maven. Apache Ant es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build).

34 Véase por ejemplo la lista extensiva de herramientas de pruebas de Danny Faught, en www.tejasconsulting.com.

35 Es una herramienta para la captura/reproducción de escenarios de una interfaz gráfica de usuario.

Apéndice A

Omninet: El Internet en Todas Partes

Documento de los Requisitos de Marketing

1. Alcance

Este documento especifica los requisitos de un grupo de quioscos para el acceso al Internet denominado Omnidet. Estos quioscos deberían proporcionar a los clientes que tengan dinero en efectivo, tarjetas de crédito o tarjetas de débito el acceso simple, rápido y fiable al Internet en lugares públicos en precios razonables por minuto de uso.

1.1 Términos, Acrónimos y Abreviaciones

Para los propósitos de este proyecto, las siguientes abreviaciones son necesarias:

AS	Servidor de Aplicaciones (<i>Application Server</i>)
Cable	Conexión de Internet de alta velocidad por cable por lo menos a 128 KBPS
CC	Tarjeta de Crédito (<i>Credit Card</i>) (para pago): American Express, Visa o MasterCard
CS	Servidor de Comunicaciones (<i>Communication Server</i>)
DBMS	Sistema de Gestión de Base de Datos (<i>Database Management System</i>)
DC	Tarjeta de Débito (<i>Debit card</i>) (para pago): PLUS o redes Cirrus.
DSL	Conexión de Internet de Alta Velocidad de Línea de Subscripción Digital (<i>Digital Subscriber Line high-speed Internet connection</i>) por lo menos a 128 KBPS
IE	El navegador de Internet Explorador de Internet (Internet Explorer)
KBPS	Kilobits por segundo
Quiosco	El punto de acceso libre e independiente a Internet de Omnidet
Linux	El sistema operativo Red Hat Linux Versión 8.0
Opera	El navegador de Internet gratuito Opera
PIN	Número de Identificación Personal (<i>Personal Identification Number</i>) (para la tarjeta de débito)
PSTN	La conexión de Internet de Red Telefónica Pública Comutada (conectividad común de acceso a internet por la red telefónica) por lo menos a 50 KBPS
URL	Localizador Universal de Recursos (<i>Universal resource locator</i>)
WS	WS Servidor Web (<i>Web Server</i>)
WXP	El sistema operativo Windows XP Professional

1.2 Documentos Aplicables

[1] Véase el Documento de los Requisitos del Sistema Omnidet para los requisitos de diseño del sistema.

[2] Véase el Documento de los Prototipos de las Pantallas de Omnidet del quiosco y el centro de llamadas (actualmente no disponible).

2. Fecha de la versión requerida

El primer conjunto de los 1.000 quioscos de Omnidet debería funcionar, aceptar los pagos y acceder al Internet en el tercer cuartal financiero.

3. Descripción de los requisitos

3.1 Requisitos técnicos generales

Omnidet debería proporcionar el acceso al Internet a los usuarios en los aeropuertos, los centros comerciales, los teatros y otros lugares públicos.

Omnidet debería proporcionar el acceso a la información a los agentes del centro de llamadas acerca de las sesiones de los quioscos actuales y anteriores, así como también la habilidad de controlar las sesiones actuales.

3.1.1 Bienvenida

Entre las sesiones, cada quiosco de Omnidet debería mostrar un mensaje de bienvenida atractivo (véase el prototipo de la pantalla K.1).

3.1.2 Pago

Una vez que el usuario navega pasada la pantalla de Bienvenida, el quiosco debería dar al usuario la opción de comprar un período de tiempo en la pantalla de Pago (Véase el prototipo de la pantalla

K.2). El quiosco debería vender períodos de tiempo en incrementos de (5) minutos, hasta (1) hora.

El sistema acepta las siguientes formas de pago:

- Efectivo (billetes solamente) (véase el prototipo de la pantalla K.3)
- Tarjeta de crédito (Sólo American Express, Visa o Mastercard) (véase el prototipo de las pantallas K.4 y K.7)
- Tarjeta de débito (Redes PLUS o Cirrus solamente) (véase el prototipo de las pantallas K.5 y K.7)

Una vez que el período actual de tiempo está dentro de los sesenta (60) segundos de la expiración, el quiosco debería mostrar un mensaje emergente que pregunta si el usuario quiere comprar más tiempo (véase el prototipo de la pantalla K.9).

3.1.3 Navegador de Internet

En la pantalla de Bienvenida, cada quiosco de Omninet debería brindar al usuario la elección de la última versión de Netscape, Opera o Internet Explorer (disponible en los quioscos con Windows solamente).

3.1.4 Rendimiento

En los quioscos que operan con una conexión PSTN, los usuarios deberían tener una velocidad de conexión mayor que 50 KBPS.

En los quioscos con conexiones DSL o de cable, los usuarios deberían tener una velocidad de conexión mayor que 128 KBPS.

3.1.5 Localización

Cada quiosco debería ser configurado para operar en el idioma local principal para su sitio instalado.

En ubicaciones donde múltiples idiomas son comúnmente utilizados, la pantalla de Bienvenida debería permitir al usuario seleccionar el idioma para la sesión.

Cada navegador del quiosco de Omninet debería ser configurado para admitir todos los idiomas compatibles con el sistema operativo y navegador.

3.1.6 Control de Contenido

Porque los usuarios de Omninet accederán a Internet en lugares públicos, Omninet debería implementar el bloqueo de los sitios para prevenir que se muestre material pornográfico, objetable, indecente, obsceno o material violento.

Omninet debería proteger cada quiosco contra el envío o el recibimiento de un virus, un gusano u otro código malicioso.

3.1.7 Terminación de la Sesión

Los usuarios pueden terminar las sesiones en una de las dos maneras:

- Cerrando la sesión (no hay reembolso para el tiempo inutilizado)
- Permitiendo que el tiempo expire.

3.1.8 Confidencialidad

Para proteger la confidencialidad del usuario—p.ej., las URLs visitadas — una vez que la sesión termine, cada quiosco debería limpiar todas las cookies y otros archivos descargados, limpiar el histórico de las URL, salir del navegador y reiniciar el navegador en la pantalla de Bienvenida.

3.2 Administración

3.2.1 Actualizaciones de Software

En circunstancias normales, las actualizaciones del software tomarán lugar automáticamente. A las 2:00 AM del horario local, cada quiosco debería conectarse a la granja de servidores y pedir actualizaciones. Esas actualizaciones incluyen:

- Parches del sistema operativo o el navegador.
- Nuevos drivers de red, modem y gráficas.
- Logos nuevos.
- Tablas actualizadas de las tasas de los pagos por minuto.
- Definiciones de virus, gusanos, código malicioso u otro cortafuego.
- Sitios Web bloqueados.

Si no hay actualizaciones disponibles, el quiosco debería desconectarse.

Si la aplicación de la actualización en el servidor de aplicaciones le comunica al quiosco que está sobrecargado, el quiosco debería desconectarse, luego reintentar después de un tiempo. El retraso

para el reintento es un período aleatorio entre diez (10) y sesenta (60) minutos.

Los agentes del centro de llamadas también pueden realizar las actualizaciones de software a los quioscos.

3.2.2 Vista de los Quioscos

Los agentes del centro de llamadas deberían ser capaces de navegar en una lista de los quioscos. Para cada quiosco, los agentes del centro de llamadas deben poder ver:

- La versión del sistema operativo actual.
- La versión actual del navegador.
- Tiempo total de funcionamiento desde la instalación.
- La actualización total desde la última actualización de software.
- El número de caídas, los reinicios u otras fallas peligrosas desde la última actualización de software.

Los quioscos deben conectarse a la granja de servidores una vez por hora para informar su estado.

Si un quiosco no se conecta a la granja de servidores, el agente del centro de llamadas puede obligar una conexión para comprobar el estado.

Si un quiosco está desconectado, ese quiosco debería mostrarse al principio de la lista de los quioscos, resaltado en rojo.

3.2.3 Vista de los Usuarios

Para aquellos quioscos que tienen usuarios activos, los agentes del centro de llamadas deberían tener acceso a la siguiente información:

- URLs actuales y anteriores.
- Número de tarjeta de crédito o débito (si es aplicable).
- Nombre (si es disponible de la validación de la tarjeta de crédito).
- Cantidad pagada para esta sesión.
- Períodos comprados de tiempo.
- Sesión anterior (si está disponible del número y nombre de la tarjeta de crédito).
- Tiempo restante pagado.

3.2.4 Modificar el Usuario

Los agentes del centro de llamadas deberían ser capaces de modificar la sesión de los usuarios añadiendo períodos de tiempo.

La anulación supervisada es necesaria para que un agente añada más de sesenta (60) minutos de tiempo por día.

3.2.5 Terminar el Usuario

Si un agente del centro de llamadas cree que un usuario ha participado en la utilización ilegal, inapropiada o fraudulenta de una sesión, el agente puede terminar esa sesión.

El usuario debería recibir un reembolso por cualquier tiempo no utilizado al momento de la terminación.

El usuario debería recibir un mensaje de que la sesión fue terminada por una actividad inapropiada. El mensaje debería especificar la cantidad del reembolso.

Apéndice B

Omninet: El Internet en Todas Partes. Documento de los Requisitos del Sistema

Requisitos funcionales del sistema.

La capacidad del sistema para proporcionar funciones, las cuales coincidan con las necesidades establecidas e implícitas, cuando el software sea utilizado con las condiciones especificadas.

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-010			<i>Adecuación</i>	
010-010-010	1.0	3.1	El diseño del quiosco debería ser adecuado para la instalación en lugares públicos.	1
010-010-020	1.0	3.1	Los quioscos deberían permitir el acceso a los sitios de Internet a través de todas las direcciones de Internet admitidas.	1
010-010-030	1.0	3.1	Los quioscos deberían proporcionar una interfaz de usuario estándar con teclado, ratón y video.	1

* Las prioridades son:

- 1 Muy alta.
- 2 Alta.
- 3 Media.
- 4 Baja.
- 5 Muy Baja.

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-010-040	1.0	3.1	Las computadoras de escritorio del centro de llamadas deberían proporcionar a los agentes una interfaz de usuario estándar con teclado, ratón y video.	2
010-010-110	1.0	3.1.2	Los quioscos deberían aceptar los pagos en moneda local.	2
010-010-210	1.0	3.1.5	Los quioscos deberían ser configurados para interactuar con los clientes completamente en el idioma local principal desde la pantalla de Bienvenida hasta el cierre de la sesión.	1
010-010-220	1.0	3.15	En la pantalla de Bienvenida, los quioscos deberían permitir a los usuarios seleccionar un idioma local secundario donde sea apropiado. Una vez que este idioma sea seleccionado, el quiosco debería interactuar con los clientes completamente en este idioma hasta el cierre la de sesión.	2

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-010-230	1.0	3.15	Todos los navegadores Web deberían poder mostrar páginas Web en Inglés (EEUU), Español (México) y Francés (Canadá) adicionalmente a todos los idiomas locales primarios y secundarios locales para los cuales el quiosco ha sido configurado.	2
010-010-240	1.0	3.15	Todos los navegadores Web del quiosco deberían poder mostrar páginas Web en hebreo, árabe, alemán, chino (mandarín), chino (Hong Kong), japonés y ruso.	3
010-010-250	1.0	3.15	Todos los navegadores Web del quiosco deberían poder mostrar páginas Web en otros idiomas aparte de los especificados en 010-010-230 y 010-010-240.	4
010-010-910	1.0	3.1.7	Los quioscos deberían permitir a los usuarios terminar las sesiones en cualquier momento haciendo clic en el ícono "Salir" en la barra de tareas.	1

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-010-920	1.0	3.1.2	En cualquier momento mientras el usuario está navegando la Web, los quioscos deberían permitir a los usuarios comprar más tiempo haciendo clic en el ícono "Comprar más Tiempo" en la barra de tareas.	3
010-010-930	1.0	3.1.2	Exactamente 60 segundos antes de que la sesión expire, los quioscos deberían presentar a los usuarios un mensaje que les dé la opción de retornar a la pantalla de pago y comprar más tiempo. El usuario puede aceptar la opción o rechazarla.	1
010-010-940	1.0	3.1.7	Los quioscos deberían terminar automáticamente las sesiones de usuario dentro del 1 segundo del tiempo de expiración del actual período de tiempo comprado por el usuario.	1
		[Más debe ser determinado]		
010-020			<i>Exactitud o Precisión</i>	

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-020-010		3.1.3	Todos los adjuntos de e-mail y páginas Web compatibles (véase sección 010-030) deberían ser visibles en los quioscos sin ninguna distorsión observable de sus fuentes en los sistemas originadores. El desplazamiento ("scrolling") no se considera distorsión.	2
			[Más debe ser determinado]	
010-030			<i>Interoperabilidad</i>	
010-030-010		3.1.3	Los quioscos deberían permitir a los usuarios mandar y recibir correos electrónicos de y a los quioscos de Omnitel.	1
010-030-020		3.1.3	Los quioscos deberían permitir a los usuarios enviar y recibir correo electrónico a y de otros computadores personales de Windows y Linux ejecutando el software de e-mail y el sistema operativo de no más de 2 años de antigüedad.	2
010-030-030		3.1.3	Los quioscos deberían permitir a los usuarios enviar y recibir correos electrónicos de y a computadoras Macintosh.	3

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-030-040		3.1.3	Los quioscos deberían permitir a los usuarios enviar y recibir correos electrónicos de y a otros sistemas tales como mainframes, servidores Unix, servidores VMS, computadores personales (Aquellos ejecutando el software de e-mail o los sistemas operativos de más de dos años de antigüedad o aquellos que no son compatibles con Windows, Linux o Mac) y otros sistemas.	5
			[Más debe ser determinado]	
010-040			Seguridad	
010-040-010	1.0	3.16	Un cortafuego que se ejecuta en el quiosco debería proteger cada quiosco contra los virus entrantes o salientes, los gusanos, otro código malicioso y una programación perfeccionista y obsesiva. En la detección de cualquier ítem como ése, el cortafuego debería informar esos eventos de seguridad al servidor de aplicaciones.	1

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-040-020	1.0	3.16	Software con función de filtro que se ejecuta en la granja de servidores, debería proteger a cada quiosco contra la entrada o salida de material pornográfico, objetable, indecente, obsceno, violento u otro material inapropiado. Al detectar cualquiera de esos ítems, el software con función de filtro debería informar al servidor de aplicación acerca de esos eventos de seguridad.	1
010-040-030	1.0	3.16	Cuando se origine un informe de un evento de seguridad, el servidor de aplicaciones debería enviar una entrada de registro al servidor de la base de datos para colocar el evento en una tabla de base de datos.	1
010-040-040	1.0	3.16	Los eventos de seguridad registrados en la tabla de la base de datos deberían ser mantenidos por lo menos un año.	3

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-040-040	1.0	3.16	Cuando se origine un informe de un evento de seguridad, el servidor de aplicaciones debería comunicar el evento a un computador de escritorio activo del agente del centro de llamadas. (Activo se define como actualmente conectado y no bloqueado). Así, cuando el agente ha sido alertado, él debería tomar acción de acuerdo con las políticas de seguridad actuales para el control de los eventos de seguridad.	1
010-040-050	1.0	3.16 3.2.5	Sujeto a las políticas de seguridad actuales para el control de los eventos de seguridad, un agente del centro de llamadas debería poder terminar la sesión de un usuario quien envía código malicioso o material inapropiado, o quien utiliza el quiosco para acceder a otros sistemas. El usuario cuya sesión es terminada, entonces debería recibir un mensaje que incluya un número gratuito al que puede llamar para apelar contra la decisión.	2

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
010-040-060	1.0	N/A	Sujeto a las políticas de seguridad actuales para el control de los eventos de seguridad, un agente del centro de llamadas debería ser capaz de entrar a su sistema de escritorio al principio de cada turno y salir al final de cada turno.	1
010-040-070	1.0	N/A	Sujeto a las políticas de seguridad actuales para el control de los eventos de seguridad, un agente del centro de llamadas debería ser capaz de bloquear su sesión de entrada actual en su sistema de escritorio antes de abandonar su escritorio.	2
			[Más debe ser determinado]	
010-050			Conformidad (Estándares/leyes/regulaciones de funcionalidad)	
			[Debe ser determinada]	

Requisitos de Fiabilidad del Sistema

La capacidad del sistema de mantener un nivel específico de rendimiento cuando es utilizado en condiciones específicas.

ID	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
020-010			<i>Madurez</i>	
020-010-010	1.0	3.1.3	Excepto a lo observado en 020-010-030, la granja de servidores debería proporcionar a los quioscos los últimos drivers, parches u otras actualizaciones para Linux o Windows (como sea apropiado), sólo después de que cada actualización haya estado disponible por lo menos 30 días, el soporte técnico haya documentado todos los asuntos y el comité del control de cambios haya aprobado la actualización.	3

ID	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
020-010-020	1.0	3.1.3	Excepto a lo observado en 020-010-030, la granja de servidores debería proporcionar a los quioscos los últimos drivers, parches u otras actualizaciones para el Internet Explorer, Netscape u Opera (como sea apropiado), sólo después de que cada actualización haya estado disponible por lo menos 30 días, el soporte técnico haya documentado todos los asuntos y el comité del control de cambios haya aprobado la actualización.	3
020-010-030	1.0	3.1.6	La granja de servidores debería proporcionar a los quioscos las últimas definiciones de los virus, los gusanos, el código malicioso y las definiciones de los sitios Web bloqueados.	1
			[Más debe ser determinado]	
020-020			<i>Tolerancia a las Fallas</i>	

ID	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
020-020-010	1.0	3.1.2	Si la conexión de la red falla durante la aprobación del pago, el quiosco debería completar la transacción si el cliente ha sido cobrado. El quiosco no debería completar la transacción si el cliente no ha sido cobrado.	1
020-020-020	1.0	3.1.2	Si la conexión de la red falla durante la aprobación del pago, el quiosco no debería completar la transacción si el cliente no ha sido cobrado.	2
			[Más debe ser determinado]	
020-030			<i>Recuperabilidad</i>	
			[Debe ser determinada]	
020-040			<i>Conformidad</i> (Estándares/leyes/regulaciones de fiabilidad)	
			[Debe ser determinado]	

Requisitos de Usabilidad del Sistema

La capacidad del sistema para ser comprendido, aprendido, utilizado y ser atractivo al usuario y los agentes del centro de llamadas cuando es utilizado en las condiciones especificadas.

ID	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
030-010			<i>Comprendibilidad</i>	
030-010-010	1.0	3.1	La interfaz del usuario del quiosco debería preservar el aspecto y la sensación de Microsoft Windows comúnmente comprendida.	2
030-010-010	1.0	3.1	La interfaz del usuario del computador de escritorio del agente del centro de llamadas debería preservar el aspecto y la sensación de Microsoft Windows comúnmente comprendida.	2
			[Más debe ser determinado]	
030-020			<i>Facilidad del Aprendizaje</i>	
			[Debe ser determinada]	
030-030			<i>Operatividad</i>	
030-030-010	1.0	3.1	La interfaz del usuario del quiosco no debería requerir que el usuario navegue a través de más de 5 páginas únicas antes de acceder al navegador Web.	2

ID	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
030-030-020	1.0	3.1	La interfaz de usuario del agente del centro de llamadas no debería exigir al usuario que navegue a través de cinco páginas únicas antes de acceder a cualquiera de las principales funciones del centro de llamadas.	2
			[Más debe ser determinado]	
030-040			<i>Atractividad</i>	
			[Debe ser determinada]	
030-050			<i>Conformidad (Estándares de usabilidad)</i>	
	1.1		El sistema debería cumplir las leyes de acceso locales para los discapacitados.	5

Requisitos de Eficiencia del Sistema

La capacidad del sistema para proporcionar el rendimiento apropiado, relativo a la cantidad de recursos utilizados en las condiciones indicadas.

ID.	Ver	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
040-010			<i>Comportamiento Respecto al Tiempo</i>	
040-010-010	1.0	3.1.1	El quiosco debería presentar la pantalla de Bienvenida dentro de los 120 segundos del encendido del quiosco el 95% de las veces. En ningún caso el tiempo debería exceder los 240 segundos.	1
040-010-020	1.0	3.1.1	El quiosco debería presentar la pantalla de Bienvenida dentro de los 15 segundos de que se haya completado la sesión anterior el 95% de las veces. En ningún caso el tiempo debería exceder los 30 segundos.	1
040-010-030	1.0	3.1.2	El quiosco debería presentar la primera pantalla de pago dentro de los 0,5 segundos cuando un cliente potencial toca la pantalla, el ratón o teclado, el 95% de las veces. En ningún caso el tiempo debería exceder los 1,5 segundos.	2

ID.	Ver	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
040-010-040	1.0	3.1.2	Excluyendo cualquier retraso del usuario y de la aprobación de una tarjeta de crédito o de débito, el quiosco debería completar el proceso del pago en no más de 15 segundos después de presentar la primera pantalla del pago el 95% de las veces. En ningún caso el tiempo debería exceder los 30 segundos. Si algún retraso de tarjeta de débito o crédito excede los 30 segundos, el quiosco debería presentar al usuario la opción de cancelar la transacción.	2
040-010-050	1.0	3.1.2	En el caso de que cualquier retraso de la aprobación de la tarjeta de débito o de crédito alcance los 60 segundos, el quiosco debería cancelar (tiempo de espera) la transacción. El usuario puede intentar la transacción.	2
[Más debe ser determinado]				
<i>040-020</i>				<i>Utilización de Recursos</i>
040-020-010	1.0	2	La granja de servidores debería admitir 1.000 quioscos que pueden estar todos activos simultáneamente.	2

ID.	Ver	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
040-020-020	1.0		La granja de servidores debería admitir 25 computadoras de escritorio para los agentes del centro de llamadas, de los cuales todos pueden estar activos simultáneamente.	2
[Más debe ser determinado]				
<i>040-030</i>				<i>Conformidad (Estándares de rendimiento)</i>
[Debe ser determinada]				

Requisitos de Mantenibilidad del Sistema

La capacidad del sistema para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones de los cambios del software en los entornos y en las especificaciones de requisitos y funcionales.

ID	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
050-010			<i>Analizabilidad</i>	
	1.0		Las organizaciones de desarrollo y mantenimiento deberían mantener la trazabilidad entre los requisitos de marketing, requisitos del sistema, riesgos de calidad, juegos de prueba y casos de prueba.	1
			[Más debe ser determinado]	
050-020			<i>Modificabilidad</i>	
050-020-010	1.1		La granja de servidores debería admitir hasta 100 agentes del centro de llamadas.	2
050-020-020	1.1		La granja de servidores debería admitir hasta 5.000 quioscos.	2
			[Debe ser determinado]	
040-030			<i>Conformidad (Estándares de rendimiento)</i>	
			[Debe ser determinada]	

Requisitos de Portabilidad del Sistema

La capacidad del sistema para ser transferido de un entorno a otro.

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
060-010			<i>Adaptabilidad</i>	
060-010-010	1.0		El hardware del quiosco debería ser compatible con Linux.	1
060-010-010	1.1		El hardware del quiosco debería ser compatible con Windows XP.	2
			[Más debe ser determinado]	
060-020			<i>Instalabilidad</i>	
060-020-010	1.0		La granja de servidores debería fomentar las actualizaciones apropiadas (para el Sistema Operativo, el navegador y el software del quiosco) para los quioscos acerca de la identificación de una versión de nivel atrasado del software en un quiosco.	1
			[Más debe ser determinado]	
060-030			<i>Co-existencia</i>	
060-030-010	1.0		Los quioscos deberían permitir la visualización de documentos de Adobe Acrobat en las páginas Web.	3
			[Más debe ser determinado]	

ID.	Ver.	MRD (Documento de Requisitos de Marketing)	Descripción	Prioridad*
060-040			Reemplazabilidad	
060-040-010	1.0		El software del servidor debería ser escrito para ser portable a través del servidor Web, servidor de aplicación y los fabricantes de servidores de bases de datos.	2
			[Más debe ser determinado]	
060-050			Conformidad	
060-050-010	1.0		El software del quiosco debería ser implementado para que funcione en cualquier navegador conforme a los estándares.	3
060-050-020	1.0		El software del servidor debería ser escrito utilizando bases de datos y lenguajes de programación estándar en la industria.	1
			[Más debe ser determinado]	

Modelos del Diseño

Arquitectura del Sistema Omninet

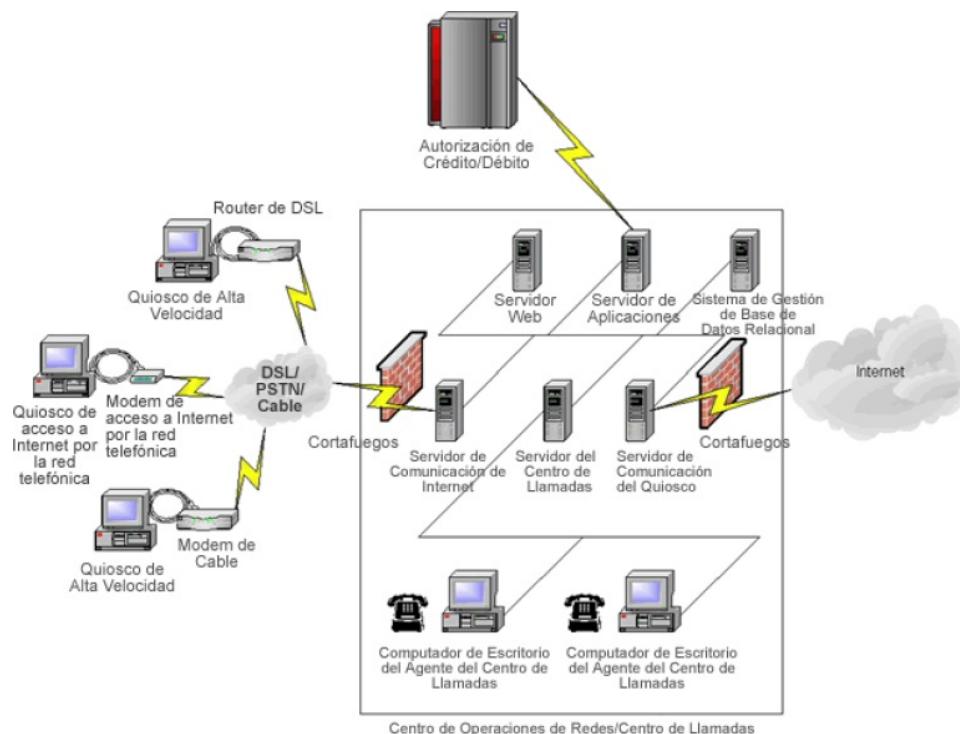


Tabla de Decisiones del Procesamiento del Pago

La siguiente tabla de decisiones muestra las reglas de negocio que rigen el proceso de los pagos. El procesamiento de la lógica de los pagos que autoriza las ejecuciones de las tarjetas de débito o crédito en el servidor de aplicaciones. El procesamiento de la lógica de los pagos que verifica las ejecuciones de las monedas legítimas o las tarjetas correctas en el quiosco.

	Regla de Negocio					
Condición	1	2	3	4	5	6
Dinero válido	No	Si	-	-	-	-
Tarjeta válida	-	-	No	Si	Si	Si
PIN válido (para débito) o PIN no requerido (para crédito)	-	-	-	No	Si	Si
Cantidad aprobada	-	-	-	-	No	Si
Acción						
Rechazar efectivo	Si	No	No	No	No	No
Rechazar tarjeta	No	No	Si	Si	No	No
Pedir cantidad menor	No	No	No	No	Si	No
Vender período de tiempo	No	Si	No	No	No	Si

Flujo del Módulo del Quiosco

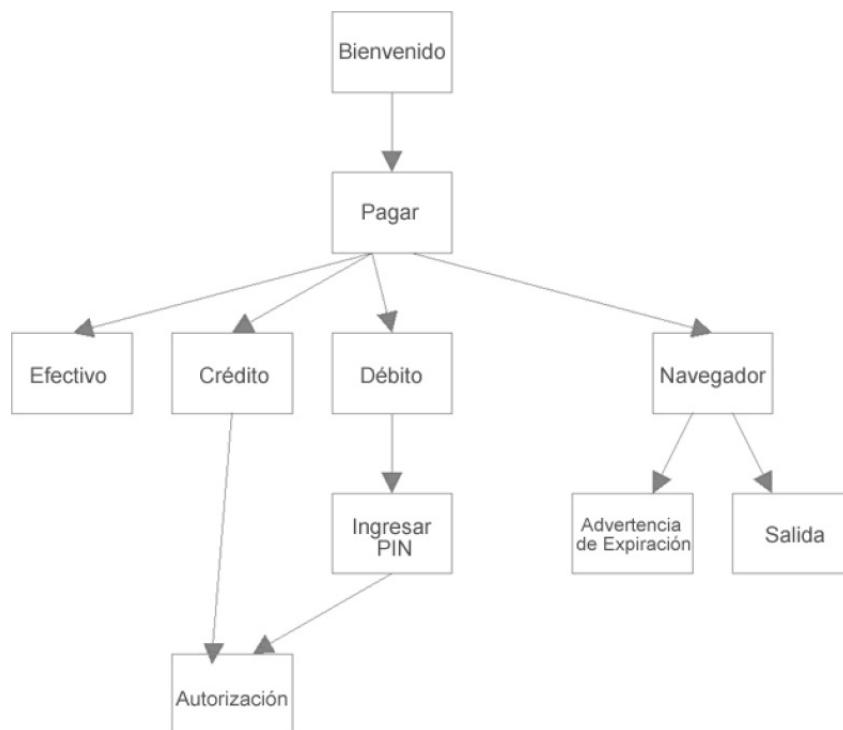


Diagrama de Transiciones de Estado del Quiosco

El siguiente diagrama muestra los estados en los que el quiosco puede estar, las transiciones que pueden ocurrir entre los estados y los eventos, las condiciones y las acciones asociadas con aquellas transiciones.

Debe ser determinado.

Tabla de Transiciones de Estado del Quiosco

La siguiente tabla muestra las transiciones de estados que pueden ocurrir basados en los eventos y las condiciones que pueden influenciar el comportamiento de los quioscos.

Nótese bien: Los siguientes eventos pueden ocurrir:

Salir El usuario termina la sesión activa del quiosco.

Terminar Un agente del centro de llamadas termina la sesión activa del quiosco.

- URL El usuario ingresa una URL (la cual puede ser navegable o estar bloqueada)
- Pagar El usuario presenta algún tipo de pago (dinero, tarjeta de crédito o tarjeta de débito) al quiosco (los cuales pueden ser válidos o inválidos) para comenzar o continuar una sesión del quiosco.

Debe ser determinado.

Estado inicial	Evento [Condición]	Acción	Estado Nuevo
...

Array Ortogonal de la Configuración del Quiosco con respecto al Sistema Operativo del Quiosco/el Navegador/la Velocidad de Conexión

El siguiente array ortogonal muestra las opciones de la configuración (únicas y en pares) que deberán ser probadas.

Debe ser determinado: Uno de estos arrays funcionará.

Configuración	Factores		
	1	2	3
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

Configuración	Factores			
	1	2	3	4
1	0	0	0	0
2	0	1	1	2
3	0	2	2	1
4	1	0	1	1
5	1	1	2	0
6	1	2	0	2
7	2	0	2	2
8	2	1	0	1
9	2	2	1	0

Configuración	Factores				
	1	2	3	4	5
1	0	0	0	0	0
2	0	1	1	1	1
3	0	2	2	2	2
4	0	3	3	3	3
5	1	0	1	2	3

6	1	1	0	3	2
7	1	2	3	0	1
8	1	3	2	1	0
9	2	0	2	3	1
10	2	1	3	2	0
11	2	2	0	1	3
12	2	3	1	0	2
13	3	0	3	1	2
14	3	1	2	0	3
15	3	2	1	3	0
16	3	3	0	2	1

Apéndice C

Soluciones de las Preguntas de Examen de Muestra y Simulación

Capítulo 1: Fundamentos de pruebas (K2)

Sección 1.1: ¿Por qué son las pruebas necesarias? (K2)

- 1. A
- 2. D
- 3. B
- 4. C
- 5. D
- 6. A

Sección 1.2: ¿Qué son las pruebas? (K2)

- 7. C
- 8. C
- 9. D
- 10. D

Sección 1.3: Principios generales de pruebas (K2)

- 11. A
- 12. C

Sección 1.4: Proceso de pruebas básico (K1)

- 13. B
- 14. D

Sección 1.5: La psicología de las pruebas (K2)

- 15. A
- 16. C
- 17. D

Sección 1.6: Código de éticas (K2)

- 18. A

Capítulo 1 Pregunta de todas las secciones

- 19. C

Preguntas del Examen de Simulación 1

- 20. A
- 21. C
- 22. B
- 23. A
- 24. B
- 25. C
- 26. A

Preguntas del Examen de Simulación 2

- 27. B
- 28. D
- 29. A
- 30. B
- 31. B
- 32. D
- 33. A

Capítulo 2: Pruebas a través del ciclo de vida de software (K2)

Sección 2.1 Modelos de desarrollo de software (K2)

- 34. D
- 35. A
- 36. C
- 37. B
- 38. C
- 39. A

Sección 2.2 Niveles de pruebas (K2)

- 40. B

- 41. B
- 42. B
- 43. D

Sección 2.3: Tipos de pruebas: los objetivos de las pruebas (K2)

- 44. B
- 45. A
- 46. C
- 47. D
- 48. A

Sección 2.4: Pruebas de mantenimiento (K2)

- 49. A
- 50. C
- 51. A
- 52. A

Capítulo 2 Pregunta de todas las secciones

- 53. A

Preguntas del Examen de Simulación 1

- 54. A
- 55. B
- 56. A
- 57. B
- 58. A
- 59. C

Preguntas del Examen de Simulación 2

- 60. C
- 61. C
- 62. D
- 63. B
- 64. A
- 65. B

Capítulo 3.0: Técnicas estáticas (K2)

Sección 3.1 Revisiones y el proceso de pruebas (K2)

- 66. A
- 67. B
- 68. D
- 69. A

Sección 3.2 Proceso de revisión (K2)

- 70. B
- 71. C
- 72. C
- 73. A
- 74. D

Sección 3.3: Análisis estático por herramientas (K2)

- 75. A
- 76. C
- 77. B
- 78. A
- 79. D

Capítulo 3 Pregunta de todas las secciones

- 80. A

Preguntas del Examen de Simulación 1

- 81. D
- 82. B
- 83. C

Preguntas del Examen de Simulación 2

- 84. A
- 85. B
- 86. D

Capítulo 4.0: Técnicas de diseño de pruebas (K3)

Sección 4.1 Proceso de desarrollo de pruebas (K3)

- 87. C
- 88. A

89. A
90. D
91. B
92. A
93. B
94. B

Sección 4.2 Categorías de las técnicas de diseño de pruebas (K2)

95. C
96. D
97. A

Sección 4.3 Técnicas basadas en la especificación o de caja negra (K3)

98. B
99. A
100. B
101. C
102. D
103. A

Sección 4.4 Técnicas basadas en la estructura o de caja blanca (K3)

104. A
105. D
106. B
107. A
108. D

Sección 4.5 Técnicas basadas en la experiencia (K2)

109. D
110. B
111. B
112. C

Sección 4.6 Selección de las técnicas de pruebas (K2)

113. D

Capítulo 4 Pregunta de todas las secciones

114. B
115. A
116. D
117. C
118. B
119. A
120. D
121. B
122. B
123. D
124. A
125. C

Preguntas del Examen de Simulación 1

126. D
127. C
128. A
129. B
130. A
131. B
132. C
133. A
134. B
135. A
136. D
137. D

Capítulo 5.0: Gestión de pruebas (K3)

Sección 5.1 Organización de las pruebas (K2)

138. B
139. A
140. B

141. D
142. B

Sección 5.2 Planificación y estimación de pruebas (K2)

143. C
144. A
145. C
146. D
147. D
148. C
149. C
150. B
151. D
152. D
153. B
154. C
155. A

Sección 5.3 Monitoreo y control del progreso de las pruebas (K2)

156. D
157. D
158. D
159. B
160. A

Sección 5.4 Gestión de configuraciones (K2)

161. C
162. B
163. D

Sección 5.5 Riesgo y pruebas (K2)

164. C
165. A
166. D
167. A
168. A
169. B

Sección 5.6 Gestión de incidencias (K3)

170. B
171. C
172. B
173. D

Capítulo 5 Pregunta de todas las secciones

174. B

Preguntas del Examen de Simulación 1

175. B
176. A
177. D
178. C
179. A
180. B
181. C
182. D

Preguntas del Examen de Simulación 2

183. C
184. B
185. C
186. A
187. D
188. C
189. D
190. C

Capítulo 6.0: Soporte de herramientas para pruebas (K2)

Sección 6.1 Tipos de herramientas de pruebas (K2)

191. B
192. A
193. C
194. D

Sección 6.2: Utilización efectiva de herramientas: beneficios y riesgos potenciales (K2)

- 195. A
- 196. A
- 197. D

Sección 6.3: Introducción de una herramienta en una organización (K1)

- 198. A
- 199. B
- 200. A

Capítulo 6 Pregunta de todas las secciones

- 201. D

Preguntas del Examen de Simulación 1

- 202. B
- 203. A
- 204. B
- 205. A

Preguntas del Examen de Simulación 2

- 206. A
- 207. D
- 208. D
- 209. A

Apéndice D

Lista de los Estándares de Pruebas

Introducción

Para complementar y extender la lista de los estándares referenciados en el Programa de Estudios Nivel Básico 2010, le proporcionamos la siguiente lista de estándares y plantillas que podrían serle útiles en sus proyectos de pruebas. Por favor note que algunos de los estándares son ampliamente observados. La utilización común de estándares es para ahorrar el tiempo al equipo de proyecto de reinventar la rueda.

Bellcore

Este estándar se relaciona a las pruebas de sistema, no con las pruebas de software, pero podría aplicarse si está probando un equipo telefónico.

GR-63-CORE: Network Equipment-Building System (NEBS) Requisitos: Protecciones Físicas

British Computer Society - Sociedad de Computación Británica

La BCS tiene múltiples estándares, pero éste se relaciona específicamente con las pruebas. Su influencia es grande porque sus conceptos y su terminología son seguidos en las certificaciones del ISEB e ISTQB.

BS-7925-2: Estándar para las Pruebas de Componente de Software

Sin embargo, éste tiene dos ventajas adicionales sobre la mayoría de los estándares citados en este documento:

1. Está disponible para su libre descarga de www.testingstandards.co.uk.
2. Incluye un conjunto muy útil de apéndices que explican muchas técnicas de pruebas comunes.

Recomendamos este estándar para su descarga y lectura.

Institute of Electrical and Electronics Engineers - Instituto de Ingenieros Eléctricos y Electrónicos

Estos son probablemente los estándares más citados comúnmente en los Estados Unidos, pero una conformidad verdadera y completa es bastante rara.

Estándar IEEE 610.12:	Glosario Estándar IEEE de la Terminología de Ingeniería de Software (nota: no es consistente con el Glosario del ISTQB)
Estándar IEEE 730:	Estándar IEEE para Planes de Aseguramiento de Calidad de Software
Estándar IEEE 730.1	Guía IEEE para Planes de Aseguramiento de Calidad de Software
Estándar IEEE 829	Estándar IEEE para la Documentación de Pruebas de Software (nota: la versión 2008 es bastante diferente que la versión 1998 actualmente seguida por el ISTQB)
Estándar IEEE 982.1	El Diccionario Estándar IEEE de Medidas para Producir Software Fiable
Estándar IEEE 982.2	La Guía IEEE para el Uso del Diccionario Estándar IEEE de Medida para Producir Software Fiable
Estándar IEEE 1008	Estándar IEEE para las Pruebas de Unidad de Software IEEE
Estándar IEEE 1012	Estándar IEEE para Planes de Verificación y Validación de Software
Estándar IEEE 1028	Estándar IEEE para las Revisiones y Auditorías de Software
Estándar IEEE 1044	Estándar IEEE para las Anomalías de Software
Estándar IEEE 1044.1	Guía IEEE para la Clasificación de Anomalías de Software
Estándar IEEE 1059	Guía IEEE para Planes de Verificación y Validación de Software
Estándar IEEE 1061	Estándar IEEE para la Metodología de Métricas de Calidad de Software
Estándar IEEE 1298	Sistema de Gestión de Calidad de Software IEEE

Si Ud. trata de seguir los estándares IEEE, considere la compra de una copia del libro de Michael Schmidt, *Implementing the IEEE Software Engineering Standards*.

International Standards Organizations - Organizaciones de Estándares Internacionales

Muchos estándares ISO podrían aplicarse al desarrollo de software y la gestión de calidad, pero sólo uno es comúnmente utilizado en las pruebas de software. Éste es un buen framework para los análisis de riesgo de calidad.

ISO Standard Estándar Internacional ISO/IEC 9126: Tecnología de información – Evaluación de Productos de Software – Características y guías de calidad de para su uso.

9126:

Software Engineering Institute - Instituto de Ingeniería de Software

Como con la mayoría de los estándares ISO, estos no son estándares de pruebas hablando estrictamente, pero ellos abordan las pruebas en algunas áreas.

CMMI: Modelo de Integración de Madurez de la Capacidad

United States Department of Defense - Departamento de Defensa de los Estados Unidos de América

Estos estándares son pocos utilizados, incluso en el trabajo del departamento de defensa de los Estados Unidos, debido a la cantidad masiva de documentos generados.

DOD-STD-2167A: Desarrollo de Software del Sistema del Ejército.

DOD-STD-2168: Programa de Calidad de Software del Sistema del Ejército

MIL-STD-480B: Control de Configuración—Cambios de Ingeniería, Desviaciones y Renuncias

MIL-STD-481B: Control de Configuración—Cambios de Ingeniería (Forma Corta), Desviaciones y Renuncias

MIL-STD-490A: Prácticas de Especificaciones

MIL-STD-499A: Gestión de Ingeniería

MIL-STD-1521: Revisiones Técnicas y Auditorías para Sistemas, Equipo, y Software de Computadoras

United States Federal Aviation Administration - Administración de la Aviación Federal de los Estados Unidos de América

Este estándar se aplica para el software de aviónica, pero también es probablemente útil para el software de seguridad crítica. Cuidado con la falta de requisitos de la cobertura de datos.

DO-178B: Consideraciones de Software en Sistemas de Aeronáutica y Certificación de Equipos.

Apéndice E

Preparación para el Examen

1. Descargue el Programa de Estudios y Glosario Nivel Básico de www.istqb.org
2. Lea el Programa de Estudios y estudie las definiciones del Glosario.
3. Rinda el primer examen de simulación de este libro. Anote cuidadosamente las áreas que necesite darle especial atención basado en las preguntas que falló.
4. Trabaje a través de cada capítulo y sección de este libro, tratando de resolver todas las preguntas de examen de muestra y todos los ejercicios. Si las preguntas de muestra y los ejercicios le indican que necesita más estudio, repase esos temas cuidadosamente hasta que los comprenda.
5. Rinda el segundo examen de simulación después de leer este libro.
6. Si pasa, debería estar listo para el examen real.
7. Si reprueba, repase los materiales en los que falló y entonces rinda el primer y segundo examen de simulación.
8. Contacte a su comité nacional acerca de las opciones de exámenes o a www.businessinnova.com.

Apéndice F

Acrónimos

API	Application Programming Interface
ASTQB	American Software Testing Qualifications Board
BD	Base de Datos
CMM(I)	Capability Maturity Model (Integration)
COTS	Commercial off-the-shelf
FFUEMP	Funcionalidad Fiabilidad Usabilidad Eficiencia Mantenibilidad Portabilidad
FMEA	Failure Modes And Effects Analysis
GUI	Graphical User Interface
HASTQB	Hispanic America Software Testing Qualifications Board
HTTPS	Hypertext Transfer Protocol Secure
IEEE	International Electronic and Electrical Engineer
IEC	International Electrotechnical Commission
ISO	International Standard Organization
ISTQB	International Software Testing Qualifications Board
K	Knowledge. (Conocimiento)
LO	Learning Objective. (Objetivo del Aprendizaje)
MRD	Marketing Requirements Document
NASA	National Aeronautics and Space Administration
OSS	Operational Support Systems
QA/QC	Quality Assurance/Quality Control
RBCS,	Inc. Rex Black Consulting Services, Incorporation
RPN	Risk Priority Number
ROI	Return On Investment
SQE	Software Quality Engineering
SRL	Sociedad Registrada Limitada
SSL	Secure Sockets Layer
SysRD	System Requirement Document
TDD	Test Driven Development
TFD	Test First Development
T-MAP	Test Management Approach
TMM(I)	Test Maturity Model (Integration)
TPI	Test Process Improvement
UCI	Unit, Component and Integration
VASIMR	Variable Specific Impulse Magnetoplasma Rocket
WBS	Work Breakdown Structure

Apéndice G

Bibliografía

- Beizer, Boris: *Software Testing Techniques*, 2e, Van Nostrand Reinhold, 1990.
- Black, Rex: *Managing the Testing Process*, 3e, John Wiley & Sons, 2010
- Black, Rex: *Pragmatic Software Testing*, 2e, John Wiley & Sons, 2007
- Buwalda, Hans et al.: *Integrated Test Design and Automation*, Addison Wesley, 2001.
- Chrissis, Mary-Beth et al.: *CMMI: Guidelines for Process Integration and Product Improvement*, Addison Wesley, 2004.
- Copeland, Lee: *A Practitioner's Guide to Software Test Design*, Artech House, 2004.
- Fewster, Mark and Graham, Dorothy: *Software Test Automation*, Addison Wesley, 1999.
- Glosario ISTQB de Términos de Pruebas disponible en www.astqb.org
- Hetzell, Bill: *Complete Guide to Software Testing*, QED, 1998.
- Kaner, Cem et al.: *Lessons Learned in Software Testing*, Wiley, 2002.
- Myers, Glenford: *The Art of Software Testing*, John Wiley & Sons, 1979.
- Programa de Estudios ISTQB Nivel Básico disponible en www.astqb.org
- van Veenendaal, Erik, ed.: *The Testing Practitioner*, UTN Publishers, 2004.

Para aumentar el valor que usted obtiene de este libro aún más, le recomendamos la siguiente literatura adicional para cada capítulo. Para obtener aún más valor de la literatura en *Pragmatic Software Testing*, asegúrese de trabajar completamente en cualquiera de los ejercicios del capítulo que ha sido recomendado.

Las referencias del libro *Managing the Testing Process* son de la tercera edición en vez de la primera o segunda. El libro *Managing the Testing Process* es publicado por Wiley en Inglés y por otros editores en China, Japón e

India/Sur Asia, y está disponible en línea a través de vendedores de libros y librerías técnicas. En algunos mercados, sólo la segunda edición está disponible actualmente, pero las diferencias de la numeración de los capítulos son pocas.

El libro *Pragmatic Software Testing* es publicado por Wiley en Inglés y por otros editores en China, Japón, e India/Sur Asia, y está disponible en línea a través de vendedores de libros y librerías técnicas. Una edición en hebreo está disponible en RBCS a pedido especial.

Capítulo 1: Fundamentos de Pruebas

Managing the Testing Process, Capítulos 1 y 8

Pragmatic Software Testing, Capítulos 1 y 4

Capítulo 2: Pruebas a Través del Ciclo de Vida de Software

Managing the Testing Process, Capítulo 11 y 12

Pragmatic Software Testing, Capítulo 3

Capítulo 3: Pruebas Estáticas

Managing the Testing Process, ninguno

Pragmatic Software Testing, Capítulo 8

Capítulo 4: Técnicas de Diseño de Pruebas

Managing the Testing Process, Capítulos 3 y 5

Pragmatic Software Testing, Capítulos 10, 12, 14, 21, y 23

Capítulo 5: Gestión de Pruebas

Managing the Testing Process, Capítulos 2 y 4

Pragmatic Software Testing, Capítulos 5 y 6

Capítulo 6: Soporte de Herramientas para las Pruebas

Managing the Testing Process, Capítulo 6

Pragmatic Software Testing, Capítulo 8

Perfil de RBCS

Por más de una decena de años, la RBCS ha dado servicios en consultoría, tercerización y entrenamiento para pruebas de software y hardware. Empleando a los más reconocidos y experimentados consultores de la industria, la RBCS conduce pruebas de productos, construye y mejora grupos de pruebas y contrata personal de pruebas para cientos de clientes alrededor del mundo. Desde las 20 primeras compañías de Fortune hasta las emergentes, los clientes de RBCS ahorran tiempo y dinero a través de un desarrollo del producto mejorado, disminución de llamadas de soporte técnico, mejora en la reputación de la compañía y más. Para obtener más información acerca RBCS visite

www.rbcs-us.com

Dirección:	RBCS, Inc. 31520 Beck Road Bulverde, TX 78163-3911 USA
Teléfono:	+1 (830) 438-4830
Fax:	+1 (830) 438-4831
E-Mail:	info@rbcs-us.com
Web:	www.rbcs-us.com

Perfil de Business Innovations

Alrededor de 7 años, Business Innovations ha dado servicios en consultoría, tercerización y entrenamiento para las pruebas de software en Europa, EEUU y Latinoamérica. Business Innovations es el primer proveedor acreditado de capacitación del ISTQB Certified Tester en Latinoamérica y es creador de herramientas para la generación de casos y datos de prueba con técnicas de caja negra, reglas de negocio y pairwise. Business Innovations conduce pruebas de productos, forma y mejora grupos de pruebas y contrata personal de pruebas para la necesidad de nuestros clientes. Los clientes de Business Innovations, ahorran tiempo y dinero a través de un desarrollo mejorado de proyecto, disminución de llamadas de soporte, mejora en la reputación de la compañía y más. Para obtener más información acerca de Business Innovations, visite www.businessinnova.com

Dirección:	Business Innovations S.R.L. Av. Landívar 205 Santa Cruz Bolivia
Teléfono	+591 (3) 3-397145
Fax:	+591 (3) 3-123320
E-Mail:	info@businessinnova.com
Web:	www.businessinnova.com

Índice Analítico

A

Acoplamiento 88
Acrónimos IX, XI, 577, 627
actividades de pruebas 10, 38, 39, 81, 82, 92, 399, 428, 493
adaptación III, 50, 91, 144
adecuación 428
administrador 223, 398, 465
administradores 112, 124, 459
Advanced 307, 310
Agrupamiento de Defectos 30
Albert Einstein 81
alfa 113
análisis de riesgos 214, 217, 498, 503, 505
análisis estático XVI, 118, 120, 163, 164, 165, 167, 168, 191, 192, 193, 194, 195, 196, 203, 204, 205, 206, 209, 319, 524, 528, 532, 533, 535, 545, 563, 575
analistas XVIII, 5, 51, 421, 485
analizador 524
arquitectura 101, 104, 105, 106, 118, 119, 142, 150, 157, 195, 260, 404, 550
aseguramiento de la calidad XV, 1, 2, 8, 23, 24, 91, 173, 391, 392, 393
Asia 630
ASTQB I, 61, 627
Ataque 256, 473
Ataque de defectos 256
ataques 123, 176, 256, 329, 337, 468, 473, 531
atributos 35, 50, 83, 390, 391, 409
automatización de pruebas XV, XVII, 397, 526, 535, 536, 538, 539, 540, 541, 542, 544, 545, 549, 553, 554, 556, 558, 559, 568, 573, 574
autor 49, 52, 53, 57, 72, 109, 169, 170, 171, 175, 176, 177, 179, 180, 199, 200, 201, 205, 206, 393, 394, 395, 471, 477

B

Backbone 106
Base de prueba 38
Bellcore 621
beneficios IX, XI, XVII, 20, 24, 54, 166, 167, 177, 191, 192, 193, 198, 204, 284, 341, 387, 388, 395, 407, 428, 465, 477, 485, 514, 523, 535, 539, 541, 553, 554, 556, 558, 565, 568, 574, 619
beneficios del análisis estático 192
beneficios típicos 191, 204
beta 113, 114, 119, 132, 141, 407, 464
Bibliografía XI, 629
Bill Hetzel 224
Boris Beizer 224
Brian Marick 329
BS-7925-2 621
bucle 108, 277, 317, 319, 320, 323, 326
buen probador 50, 72
bug 2, 4, 66, 475
Business Innovations I, 52, 110, 172, 398, 558, 633

C

Calidad XVIII, 2, 24, 128, 130, 147, 215, 219, 223, 226, 237, 244, 392, 393, 395, 434, 622, 623
caminos 21, 136, 192, 205, 260, 299, 317, 321, 323, 324, 468
capacidad 1, 2, 7, 43, 55, 98, 107, 108, 109, 113, 126, 127, 131, 132, 165, 223, 225, 252, 289, 334, 350, 376, 379, 390, 404, 411, 439, 502, 533, 543, 585, 594, 597, 599, 602, 603
capacitación 61, 92, 176, 409, 424, 463, 495, 552, 554, 556, 557, 568, 572, 633
Capers Jones 25, 28, 29, 52, 168, 172, 196, 394
Carl Edward 523
cartas de pruebas 328, 329, 330, 332, 333, 468

Caso de Estudio 333, 402, 547
Caso de prueba 14
Caso de uso 285
casos de prueba XVI, XVII, 14, 15, 16, 38, 39, 44, 45, 48, 50, 75, 99, 124, 146, 152, 155, 164, 168, 172, 178, 211, 212, 214, 217, 219, 229, 230, 231, 233, 248, 249, 255, 257, 258, 276, 284, 285, 286, 298, 300, 301, 307, 312, 313, 319, 327, 336, 337, 338, 342, 343, 350, 351, 352, 353, 361, 362, 367, 368, 375, 379, 384, 390, 397, 398, 404, 406, 417, 419, 423, 427, 432, 434, 435, 437, 438, 439, 443, 447, 451, 455, 468, 480, 481, 490, 492, 495, 496, 519, 528, 537, 556, 557, 565, 602
categorías 215, 217, 220, 223, 224, 225, 227, 256, 258, 261, 418, 434
causa raíz 1, 13, 14, 37, 64, 77
Certified Tester XV, 633
ciclo de vida X, XVIII, 4, 8, 9, 10, 15, 24, 25, 41, 42, 54, 68, 81, 82, 83, 84, 85, 90, 91, 92, 93, 95, 115, 116, 117, 118, 131, 134, 144, 146, 156, 158, 159, 163, 177, 218, 340, 389, 403, 408, 411, 412, 415, 418, 428, 455, 476, 477, 478, 491, 493, 523, 612
Ciclo de vida del desarrollo 340, 344
Cierre 41, 46
clases de equivalencia 263, 265, 286, 287, 288
CMMI 92, 130, 145, 181, 202, 566, 623, 629
cobertura XVI, XVII, 7, 39, 42, 44, 47, 71, 78, 108, 110, 114, 117, 129, 140, 141, 147, 212, 231, 232, 233, 234, 246, 256, 257, 259, 260, 271, 272, 280, 284, 285, 286, 304, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 324, 326, 328, 329, 330, 332, 335, 339, 340, 341, 342, 344, 352, 357, 359, 360, 361, 362, 363, 368, 375, 384, 385, 412, 420, 423, 427, 434, 435, 444, 448, 452, 462, 492, 495, 496, 498, 531, 534, 536, 544, 549, 563, 573, 624
Cobertura 39, 141, 232, 237, 251, 257, 316, 318, 359, 367, 434
cobertura de bucle 317, 319
cobertura de código 78, 141, 147, 232, 313, 314, 315, 319, 340, 344, 360, 531
cobertura de condición y decisión modificada 317
cobertura de decisión 316, 359, 360, 361, 363, 375, 385
Cobertura de pruebas 39
cobertura de rama 316, 317, 318, 319, 324, 326
Cobertura de sentencia 257, 367
código de éticas 60, 61
Comparador de pruebas 530, 570, 573
Comparadores de pruebas 534
Compilador 195, 203
Complejidad 165, 193, 322, 543
complejidad ciclomática 165, 193, 320, 321, 322, 324
completitud 108, 174, 179, 313, 415, 420
comportamiento 4, 5, 14, 43, 97, 101, 108, 111, 112, 115, 121, 122, 128, 129, 136, 165, 167, 192, 253, 255, 256, 285, 286, 287, 288, 307, 309, 311, 314, 327, 332, 349, 358, 373, 377, 379, 395, 400, 423, 424, 470, 474, 475, 476, 498, 515, 575, 608
concentración 56
conceptos clave 9, 38, 49, 60, 81, 96, 120, 133, 163, 169, 191, 212, 255, 285, 313, 327, 339, 388, 406, 426, 454, 460, 466, 523, 535, 551
condición 2, 7, 14, 53, 93, 193, 211, 273, 275, 276, 277, 303, 304, 307, 309, 310, 316, 317, 318, 319, 357, 373, 379, 382, 389, 431
confianza 3, 7, 9, 32, 37, 52, 54, 55, 67, 74, 96, 101, 108, 109, 113, 119, 145, 160, 178, 229, 255, 256, 288, 329, 330, 334, 335, 386, 391, 412, 427, 428, 519, 521
confirmación XVI, 13, 14, 15, 38, 45, 51, 70, 75, 78, 119, 120, 121, 129, 131, 132, 145, 151, 155, 157, 160, 299, 428, 521, 527, 543
conformidad 7, 14, 53, 61, 62, 91, 113, 164, 170, 180, 192, 194, 390, 454, 532, 622
conjunto de pruebas 7, 17, 99, 142, 160, 283, 312, 317, 321, 373, 513, 531
constructor 57, 58
contexto XVI, 10, 11, 12, 35, 36, 60, 81, 88, 95, 138, 213, 246, 339, 413, 420, 474, 479
contingencia 39, 425, 461
Contingencia 464, 465
control VIII, XI, XIII, XIV, XVII, 21, 22, 38, 41, 42, 47, 69, 75, 98, 192, 203, 221, 225, 241, 243, 245, 249, 261, 279, 292, 303, 313, 314, 315, 316, 320, 321, 356, 363, 378, 387, 388, 389, 390, 391, 392, 408, 416, 418, 426, 428, 456, 459, 462, 465, 466, 471, 479, 483, 486, 489, 496, 500, 501, 505, 526, 533, 535, 537, 539, 544, 547, 553, 559, 570, 571, 592, 593, 594, 595, 617
control de calidad XIV, 389, 390, 391
Control de pruebas 426
criterios de continuación 411, 412
criterios de entrada 39, 40, 116, 171, 174, 175, 180, 181, 254, 410, 420, 423, 487

Criterios de entrada 40, 200, 501
criterios de salida 15, 38, 39, 41, 42, 46, 48, 70, 78, 175, 313, 406, 412, 413, 415, 419, 423, 426, 427, 428
Criterios de salida 38, 406
cronograma XVI, 23, 35, 39, 42, 85, 117, 187, 212, 213, 345, 397, 406, 409, 411, 413, 416, 417, 428, 448, 452, 463, 489, 490, 491, 497
cronogramas 23, 59, 84, 85, 166, 417, 559

D

Danny Faught 550
daño XV, 1, 2, 23, 57, 58, 138, 219, 249, 460, 465
Datos de prueba 39
Dave Parnas 211
David Gelperin 421
David Rico 166
decisión XVII, 142, 159, 186, 247, 257, 270, 271, 272, 273, 275, 284, 301, 302, 303, 304, 305, 306, 307, 313, 316, 317, 318, 344, 357, 359, 360, 372, 382, 469, 498, 499, 522, 551, 592
Defecto 2, 34, 66, 476
defectos abiertos 430, 441, 445, 449
defectos pospuestos 430
Densidad de Defectos 413
Depuración 9
desarrollador 395, 398, 472, 521
Desarrolladores 466
desarrollo de software VI, X, 5, 9, 40, 81, 82, 87, 94, 99, 143, 164, 222, 538, 551, 556, 612, 623
Desarrollo dirigido por pruebas 99
desastre 332, 392, 453, 556
destructor 57, 58
desventajas XVII, 331, 387, 388
Diagrama X, 278, 308, 608
diferencias XVI, 152, 163, 167, 169, 255, 288, 343, 405, 429, 491, 630
dinámicas XVI, 15, 53, 90, 120, 121, 163, 164, 167, 168, 191, 193, 197, 203, 209, 314, 331, 332, 333, 341, 389, 421, 528, 530, 548
dirección 48, 56, 60, 61, 64, 299, 300, 395, 411, 454, 554, 558, 560, 561, 562
dirigidas por datos XVII, 99, 571
dirigidas por palabras clave XVII, 549
diseño de pruebas XI, XVII, 12, 38, 43, 47, 49, 51, 93, 121, 152, 165, 197, 212, 213, 214, 218, 234, 256, 259, 260, 284, 285, 309, 311, 315, 319, 327, 328, 331, 339, 343, 352, 353, 365, 374, 376, 377, 379, 401, 419, 420, 462, 480, 487, 528, 529, 533, 537, 563, 615
DO-178B 624
documentación 4, 38, 40, 109, 133, 159, 164, 169, 212, 214, 229, 231, 245, 256, 337, 342, 343, 344, 350, 364, 408, 418, 435, 439, 458, 462, 465, 474, 479, 481, 493
Documento de los Requisitos de Marketing de Omninet 182, 189, 233, 245
Documento de los Requisitos del Sistema Omninet 233, 234, 271, 344, 578
DOD-STD 623
dos pares de ojos 178
Driver 98

E

E.E.U.U. II, XIII
efecto 12, 73, 78, 117, 219, 269, 525, 563, 564, 570
efectos XV, 1, 2, 64, 122, 129, 140, 223, 224, 284
Eficiencia X, 223, 225, 599, 627
ejecución de pruebas XVII, 15, 32, 38, 116, 212, 213, 272, 327, 328, 331, 345, 406, 492, 496, 497, 505, 519, 524, 529, 530, 533, 534, 535, 539, 544, 559, 563, 566, 575
Ejercicios V, VI, VII, VIII, IX, 16, 36, 47, 59, 93, 118, 131, 142, 167, 181, 195, 233, 258, 324, 336, 343, 403, 422, 440, 459, 464, 481, 533, 549, 560
Elisabeth Hendrickson 329
encriptación 124
enfoque 55, 59, 92, 331
enmascaramiento de defectos 78
Entorno de pruebas 109

entrenamiento 222, 556, 631, 633 equipo XVII, 14, 15, 21, 36, 42, 48, 51, 52, 53, 58, 60, 72, 74, 113, 171, 198, 221, 225, 228, 229, 330, 333, 338, 342, 364, 365, 387, 388, 389, 390, 391, 392, 393, 395, 396, 398, 399, 401, 402, 403, 404, 405, 417, 419, 420, 424, 440, 441, 444, 445, 448, 458, 459, 462, 463, 465, 467, 474, 477, 480, 484, 485, 487, 494, 497, 498, 500, 502, 504, 507, 511, 513, 517, 518, 521, 546, 548, 551, 552, 556, 565, 621
equipos de pruebas 22, 36, 52, 135, 389, 394, 395, 398, 403, 474, 484
equipos independientes 394
Erich Gamma 118
Error 2, 66, 266, 267, 268, 281
errores típicos 191
escalación 462
escenarios 63, 89, 170, 179, 260, 271, 280, 285, 300, 301, 369, 440, 543, 550
Escribano 171
Esencialidad 88
especificación del diseño de pruebas 211, 213, 229, 368, 378, 406, 419, 435
estabilidad 415, 542, 552
Estándar IEEE 146, 200, 202, 209, 345, 350, 351, 426, 466, 487, 495, 496, 499, 500, 501, 506, 508, 622, 623
Estándares de Pruebas XI, 621
estáticas VII, X, XVI, 15, 120, 163, 164, 165, 167, 197, 209, 366, 377, 386, 389, 530, 548, 573, 614
estimación de pruebas VIII, XI, 387, 461, 487, 617
estrategia de pruebas 138, 139, 214, 223, 309, 330, 331, 333, 419, 423, 486, 495
Estrategia de pruebas 39, 397
estrategias 42, 129, 136, 137, 141, 217, 231, 232, 327, 331, 332, 333, 341, 396, 409, 418, 419, 420, 421, 423, 428, 512, 542
estrategias de pruebas 137, 141, 231, 327, 331, 341, 418, 419, 420, 421, 542
estrés 112, 118, 126, 131, 147, 149, 534, 559, 563
estructura detallada del trabajo 416, 417, 424
Europa 633
evaluación XVII, XVIII, 7, 15, 38, 39, 42, 46, 48, 53, 63, 68, 73, 78, 110, 120, 122, 133, 159, 163, 170, 173, 175, 199, 201, 222, 227, 228, 272, 298, 317, 389, 390, 394, 395, 419, 426, 483, 486, 501, 532, 551, 552, 568
exactitud 8, 477
Examen de Simulación XI, 614
éxito XV, 38, 48, 49, 51, 55, 59, 70, 151, 155, 243, 368, 373, 453, 544, 545, 548, 551, 553, 554, 556, 559, 560, 568
expectativas 2, 5, 15, 22, 23, 77, 110, 124, 286, 314, 390, 418, 463, 530, 554, 557, 561, 562, 573
experticia XIV, XVII, 171, 407, 504, 545, 561
exploratorias 19, 78, 256, 261, 328, 333, 334, 335, 337, 365, 423, 428, 462, 468, 498, 509, 529, 561

F

factores de éxito 555
falacia 20, 35, 37, 69, 74, 76
Falla 2, 66, 223, 225, 251, 463
fallas XIII, 3, 5, 6, 9, 13, 52, 53, 57, 72, 76, 96, 126, 129, 167, 203, 205, 209, 220, 221, 225, 256, 261, 352, 363, 411, 420, 463, 468, 469, 470, 475, 476, 485, 496, 499, 500, 509, 511, 544, 553, 581
Falta 2, 463
fases XVI, XVIII, 9, 25, 81, 85, 91, 116, 118, 141, 166, 213, 410, 418, 419, 440, 545
fases de pruebas 25, 81, 118, 166
Fiabilidad X, 108, 125, 132, 223, 225, 241, 252, 260, 594, 627
Flujo de control 192
Flujo de datos 192
Foundation Level XV
Framework de pruebas unitarias 559
frameworks 40, 394, 528, 530, 533, 539, 544, 545, 556, 563
Franklin Chang 1
Fred Brooks 174

G

Gary Rueda Sandoval I, II, III, XIV
gerencia 10, 42, 43, 54, 71, 135, 141, 178, 206, 453, 462, 463, 471, 485, 507
gestión de pruebas 53, 233, 388, 408, 426, 454, 486, 526, 533, 535, 546, 559, 563, 564, 573

Gestionado 92, 145
Gestionado cuantitativamente 92, 145
Glenford 16, 629
Glenford Myers 16
Glosario del ISTQB2, 6, 9, 11, 14, 15, 20, 38, 49, 63, 82, 83, 85, 87, 90, 93, 97, 98, 99, 108, 109, 112, 113, 121, 126, 127, 133, 141, 164, 165, 166, 169, 170, 171, 192, 195, 212, 213, 219, 231, 256, 257, 284, 316, 328, 343, 387, 390, 397, 408, 413, 419, 426, 454, 466, 469, 483, 524, 526, 529, 530, 531, 535, 559, 622
gorila afeitado 342, 408
GR-63-CORE 621
grado de cobertura 221, 222, 231, 232, 234, 245, 257, 259, 315, 423
grado de cobertura de las pruebas 222, 234, 245, 423
grados de independencia 54
granularidad 115
Greg Kubaczkowski 549

H

habilidad 51, 72, 95, 149, 174, 212, 231, 315, 327, 333, 342, 347, 385, 467, 470, 538, 565, 578
habilidades XV, 50, 51, 52, 53, 58, 59, 60, 78, 102, 115, 230, 398, 399, 400, 401, 418, 420, 462, 463, 473, 545, 552, 554, 557, 561
HASTQB I, 627
Herramienta de frameworks de pruebas unitarias 559
Herramienta de pruebas de seguridad 531
Herramienta de seguridad 531, 573
Herramientas de seguridad 534

I

IEEE 1028 180, 200, 202, 209
IEEE 12207 91, 92, 130, 146, 181
IEEE 829 202, 229, 230, 231, 345, 350, 351, 408, 409, 419, 421, 422, 426, 435, 436, 439, 458, 460, 474, 475, 479, 481, 487, 495, 496, 500, 506, 508, 515, 516
Impacto 508, 519
implementación 10, 19, 24, 25, 27, 28, 38, 42, 44, 45, 48, 73, 75, 78, 83, 84, 92, 98, 99, 101, 139, 164, 182, 199, 214, 218, 228, 234, 265, 271, 311, 318, 386, 394, 395, 396, 397, 419, 454, 456, 470, 479, 483, 495, 501, 539, 551, 554, 570
imposibilidad 20, 389
Incidencia 38
Independencia 49
Independencia de pruebas 49
informe de defecto XVII, 14, 178, 438, 466, 467, 468, 469, 470, 471, 472, 473, 474, 476, 481, 482, 507, 510, 516, 519
Informe de pruebas 426
Informe del resumen de las pruebas 426
Informe del resumen de pruebas 39
informes 14, 25, 38, 41, 43, 45, 46, 48, 50, 51, 61, 73, 111, 172, 178, 224, 234, 330, 397, 398, 408, 409, 423, 426, 427, 428, 434, 435, 438, 441, 445, 449, 458, 459, 466, 467, 468, 469, 470, 473, 474, 476, 477, 478, 481, 508, 524, 526, 527, 530, 533, 546, 548, 553, 559, 565
ingeniero de pruebas 402, 404, 561
instalabilidad 502
integración 83, 87, 94, 95, 96, 100, 102, 103, 104, 105, 106, 107, 117, 118, 119, 146, 158, 168, 194, 245, 259, 374, 394, 403, 423, 428, 448, 453, 464, 497, 539, 543, 548
interesados del negocio 3, 23, 35, 38, 46, 54, 73, 97, 135, 167, 214, 220, 222, 224, 225, 227, 228, 298, 377, 418, 421, 425, 460, 461, 475, 520, 557
Interoperabilidad 113, 589
introducción de una herramienta XVII, 551, 567
Irremplazabilidad 88
ISO 9126 121, 130, 147, 151, 181, 202, 223, 224, 225, 285, 314
ISTQB I, XIV, XV, 13, 22, 41, 42, 47, 61, 63, 91, 92, 116, 120, 121, 143, 171, 195, 197, 220, 317, 330, 343, 345, 389, 481, 484, 563, 621, 622, 627, 629, 633

J

James Bach 329
James Whittaker 329

Jefe de Pruebas 390, 403, 546
Jenny Stevens 298
John Wiley 629
Juego de prueba 39
juegos de pruebas 34, 124, 135, 139, 217, 219, 397, 416, 437, 513

K

Kent Beck 118

L

Latinoamérica 633
líder. XVII, 60, 65, 171, 200, 387, 388, 396, 400, 484, 485, 486, 548
líderes 62, 169, 392, 396, 480, 510
listas de comprobación 108, 170, 171, 256, 328, 330, 332, 420, 528

M

madurez 40, 41, 92, 145, 418, 548, 551
malas noticias 59, 473
Mantenibilidad X, 127, 223, 225, 602, 627
mantenimiento. XVI, 3, 9, 11, 12, 29, 46, 47, 48, 64, 75, 82, 91, 94, 95, 111, 112, 127, 133, 134, 135, 147, 153, 154, 155, 194, 201, 337, 341, 423, 424, 487, 536, 541, 542, 543, 544, 553, 554, 555, 565, 602
Mark Twain 35
MC/DC 317
McCabe 320, 321, 322, 323, 324
mejoramiento 8, 72, 92, 171, 175
mentalidad 49, 51, 72, 74, 75, 78, 93, 178
metodologías ágiles 52
métodos ágiles 86
métodos de pruebas 352, 419
métrica ciclomática 321
métricas XVII, 55, 166, 171, 175, 176, 177, 193, 201, 204, 320, 406, 407, 408, 415, 417, 426, 427, 428, 429, 436, 438, 489, 492, 496, 518, 526, 528, 559, 573
Michael Schmidt 623
Michelle Egli III
MIL-STD 624
mitigación 328, 331, 335, 425, 435, 461, 540
Modelo de desarrollo incremental 85
Modelo de desarrollo iterativo 85
Modelo V 82, 156
modelos de desarrollo XVI, 81, 144
monitoreo 63, 388, 408, 426, 440, 489, 492, 496, 524, 527, 531, 563, 568
Monitoreo de las pruebas 408
monkeys 549
mono mareado 468
monos tontos 549
monos tontos probando 549
multicondición 316, 317

N

necesidad de las pruebas XV, 2
necesidades XIV, 2, 20, 35, 36, 66, 70, 77, 83, 114, 228, 340, 341, 342, 390, 391, 405, 409, 466, 495, 527, 552, 561, 585
nivel alto 213, 229, 250, 391
nivel bajo 214, 218, 219
Nivel Básico XV, 1, 63, 143, 171, 180, 181, 197, 202, 256, 345, 374, 481, 484, 525, 526, 563, 621, 625, 629
nivel de los riesgos 97, 132, 397, 413
nivel de riesgo XVI, 71, 140, 221, 222, 226, 227, 228, 272, 293, 328, 331, 503, 522
niveles. XIV, XVI, XVII, 24, 39, 40, 48, 53, 83, 92, 93, 96, 97, 99, 107, 109, 110, 111, 115, 116, 117, 118, 119, 120, 131, 145, 150, 157, 158, 161, 164, 168, 169, 177, 193, 221, 222, 226, 229, 234, 250, 313, 315, 319, 324, 326, 339, 374, 389, 390, 393, 394, 395, 397, 398, 406, 407, 408, 418, 419, 534

niveles de independencia 53, 393
Niveles de pruebas VI, X, 146, 612

O

Objetivo de prueba 11
objetivos claros 49, 176
Objetivos de las pruebas 339, 344
objeto de prueba 97, 101, 109, 111, 117, 124, 164, 167, 214, 218, 256, 390, 410, 455, 457, 458, 475, 480, 481, 486, 505
objeto de pruebas 111, 117, 124, 167, 214, 218, 457, 475, 480, 483
Omninet IX, X, 94, 95, 118, 119, 142, 167, 168, 181, 182, 185, 195, 196, 233, 234, 237, 244, 250, 258, 262, 273, 275, 276, 343, 403, 404, 405, 422, 423, 459, 464, 481, 533, 561, 577, 578, 579, 580, 585, 589, 605
operaciones 5, 9, 108, 165, 199, 227, 252, 392, 444, 445
optimización 12
orientación XV, 50, 553, 556

P

pairwise 124, 633
paradoja del pesticida 20, 31, 32, 34, 76, 79
pares 11, 66, 69, 101, 124, 169, 171, 172, 178, 180, 200, 205, 206, 277, 307, 309, 310, 398, 474, 533, 609
Particionamiento de equivalencia 284, 367
Paul Gerrard 224
Perfil de Business Innovations XI, 633
Perfil de RBCS XI, 631
pervasive testing 115
piloto 113, 141, 538, 551, 555, 558, 559, 560, 568, 569, 573, 574
plan de pruebas 62, 168, 196, 333, 397, 406, 407, 408, 409, 411, 419, 422, 424, 435, 460, 481, 494, 495, 502
Plan de pruebas 39, 487
planificación 15, 38, 39, 42, 46, 47, 68, 70, 71, 75, 83, 84, 175, 221, 331, 364, 388, 389, 406, 407, 408, 409, 413, 417, 420, 456, 460, 462, 479, 480, 489, 491, 492, 493, 505, 518, 524, 539, 556, 559
plantilla 226, 227, 229, 230, 350, 351, 408, 422, 435, 436, 439, 458, 460, 474, 495, 499, 508
Plantilla 262
Portabilidad X, 108, 223, 225, 566, 603, 627
predicción 328, 376
prediseñadas 261, 327, 336, 338
Preguntas del Examen V, VI, VII, VIII, IX, X, XI, 74, 77, 156, 159, 206, 208, 368, 378, 510, 517, 570, 573, 612, 613, 614, 616, 618, 619
Presencia de Defectos 20
Pretty Tight XIII
principios XV, 20, 36, 37, 39, 551
Probador XV, 17, 63, 143, 197, 274, 345, 387, 401, 484, 563
Probadores 467
Problemas de calidad del fabricante 88
Procedimiento de prueba 39
procedimientos de prueba XVI, 45, 212, 213, 214, 219, 231, 329, 343, 397, 406, 408, 435, 480, 481
proceso de desarrollo de pruebas VII, 212, 345 proceso de pruebas VII, X, XV, 12, 14, 38, 40, 41, 42, 47, 73, 97, 98, 108, 113, 127, 161, 163, 197, 221, 229, 332, 418, 419, 420, 421, 479, 486, 510, 514, 523, 526, 536, 539, 551, 553, 556, 559, 567, 614
proceso de pruebas básico 41, 42, 479, 523
proceso de revisión 170, 171, 176, 178, 528, 559
procesos de apoyo 91
procesos de pruebas XV, 40, 47, 92, 418, 553, 556
productos del trabajo 15, 81, 157, 163, 178, 197, 332, 346, 389, 467, 527, 537, 553
profesionalismo XV, 60
programación de pares 169
programadores XVII, 5, 8, 9, 12, 13, 14, 51, 52, 53, 65, 86, 98, 99, 102, 131, 149, 158, 194, 204, 249, 291, 319, 396, 398, 399, 523, 547, 548
prueba contra las versiones anteriores 470
prueba de comportamiento 121
prueba de concepto XVII, 551, 552

prueba de rendimiento 162
prueba de seguridad 124
prueba de sistema 24, 119, 132, 156, 164, 195, 254, 394, 407, 410, 412, 415, 423, 465, 513
prueba de todas las combinaciones 317
prueba estructural 121, 148
prueba funcional 148, 162
prueba no funcional 148, 162
prueba para cada nivel XVI, 96
prueba para el mantenimiento 542
prueba para evaluar los resultados 121
prueba relacionada con el cambio 148
pruebas ágiles 98
pruebas aleatorias 76, 549
Pruebas alfa 113
pruebas automatizadas 45, 140, 369, 419, 423, 536, 537, 539, 541, 542, 543, 550, 551, 554, 556, 557, 558, 559, 560, 561, 566, 571
Pruebas basadas en la estructura 257
Pruebas basadas en los riesgos 97
Pruebas beta 114, 261
Pruebas de caja blanca 97
Pruebas de caja negra 121, 147
Pruebas de campo 114
Pruebas de carga 112
Pruebas de casos de uso 285
pruebas de componente 79, 83, 88, 96, 97, 98, 149, 161, 162, 194, 313, 359, 419
Pruebas de componente 83
Pruebas de fiabilidad 108
pruebas de humo 462
pruebas de integración 11, 24, 25, 27, 28, 51, 83, 89, 95, 101, 102, 104, 107, 112, 113, 118, 119, 131, 150, 156, 157, 162, 315, 350, 389, 406, 419, 533
Pruebas de integración 83
Pruebas de interoperabilidad 113
Pruebas de mantenibilidad 127
Pruebas de mantenimiento. VI, X, 81, 133, 153, 160, 613
Pruebas de mono 543
pruebas de pares 169
Pruebas de Portabilidad 108
Pruebas de rendimiento 97, 153
Pruebas de robustez 93
Pruebas de seguridad 108, 149, 153
pruebas de sistema 11, 27, 28, 59, 77, 83, 84, 90, 107, 108, 109, 110, 111, 112, 115, 119, 131, 177, 193, 389, 403, 419, 487, 492, 497, 621
Pruebas de sistema 83
pruebas de unidad 28, 83, 96, 97, 98, 99, 107, 115, 118, 194, 319, 324, 389, 392, 394, 395, 396, 398, 410, 423, 470, 514, 548, 559
Pruebas de usabilidad 109, 149
pruebas dirigidas por datos 535, 544
pruebas dirigidas por palabras clave 535
pruebas dominantes 115, 116, 117, 134, 167, 418
Pruebas Dominantes 114
Pruebas estructurales 97, 120
Pruebas exhaustivas 20
Pruebas exploratorias 328, 363
pruebas falladas 513, 539
pruebas imprecisas 342
pruebas independientes XVII, 49, 86, 387, 388, 393, 394, 395, 396, 484, 485, 511
pruebas informales 330
Pruebas negativas 256
pruebas operacionales 112, 146, 161
pruebas operativas 11
pruebas pasadas 250, 426
pruebas piloto 11, 113, 114
pruebas precisas 335, 342

pruebas prediseñadas 338
Pruebas superficiales 234
pruebas tempranas 20, 23, 24, 37, 75, 76, 79, 103, 167, 177, 395, 459
psicología de las pruebas V, X, 71, 611

Q

QA XVIII, 8, 392, 393, 550, 627
QC XVIII, 627

R

ramas 21, 257, 316, 318, 320, 322, 324, 531
RBCS I, 40, 52, 61, 110, 172, 299, 398, 402, 558, 627, 630, 631
reactivas 336, 418, 512
Red Central 106
Registro de incidencias 483, 506
Registro de pruebas 39
Rendimiento IX, 97, 125, 131, 242, 253, 261, 543, 579
Requisito 2, 83
Requisito funcional 83
Requisito no funcional 83
Requisitos IX, X, 94, 118, 181, 182, 230, 234, 251, 252, 276, 277, 278, 339, 344, 463, 577, 578, 585, 594, 597, 599, 602, 603, 621
Requisitos del Sistema X, 118, 181, 251, 252, 276, 277, 278, 585
resultado esperado 16, 75, 268, 286, 287, 314, 368, 475, 499, 507
resultado real 122, 220, 461, 495
Resumen 436, 437, 438, 482, 519
Revisión VI, 63, 166, 169, 171, 174
revisión de código 66, 118
revisión entre pares 170
revisión por pares 169, 172, 200, 201, 205
Revisor 170, 200, 207
Rex Black I, II, 16, 61, 134, 181, 186, 224, 233, 247, 262, 271, 276, 304, 324, 333, 334, 403, 534, 627
Richard Feynmann 163
Rick Craig 224, 557
Riesgo VIII, XI, 6, 237, 387, 460, 464, 465, 483, 502, 618
Riesgo alto 464
riesgo de calidad 221, 223, 225, 227, 483, 623
Riesgo de producto 483, 502
Riesgo de proyecto 483
Riesgo medio 465
riesgos de calidad XVI, 6, 7, 43, 84, 96, 101, 108, 111, 138, 140, 213, 214, 215, 220, 221, 223, 224, 225, 226, 227, 228, 233, 235, 245, 250, 272, 315, 321, 328, 329, 330, 331, 390, 422, 423, 424, 425, 434, 435, 554, 602
riesgos de producto 220, 460
riesgos de proyecto XVII, 84, 220, 460, 461, 463
Risk Priority Number 627
Robustez 93

S

Seguridad 109, 123, 131, 252, 260, 590
selección XVII, 40, 56, 101, 152, 174, 185, 240, 247, 293, 339, 343, 420, 551, 553, 556, 557, 559, 561, 568
selección de las técnicas XVII, 339, 343, 420
sentencia XVII, 2, 108, 193, 313, 316, 317, 318, 319, 325, 326, 359, 360, 522
similitudes 167, 429
simulador 195
Soluciones X, 611
soporte de herramientas XVII, 535, 551
Stub 98
subcaracterísticas 130, 223
switch-coverage 310
Syllabus XV, 1

T

tablas de decisión XVI, 284, 285, 301, 307, 309, 339, 354, 360
Tasa de fallas 469
taxonomía 120, 121, 329, 420, 468
técnicas de diseño de pruebas VII, XI, XV, XVII, 39, 211, 255, 297, 300, 313, 318, 339, 352, 419, 615
Técnicas de Integración 103
técnicas de pruebas VIII, XI, 32, 108, 116, 211, 221, 258, 279, 327, 328, 330, 333, 339, 344, 366, 386, 616, 621
técnicas dinámicas 366, 386
técnicas estáticas XVI, 163, 167, 206
técnico de pruebas 334
técnicos de pruebas 333, 334, 335, 398, 404
tempranas 10, 204, 214, 218, 254, 396, 424
tercerizar 462
test charters 328, 332
Test Driven Development 99, 628
Test First Development 99, 628
Test Management Approach 628
Test Maturity Model (Integration) 628
Test Process Improvement 628
Testware 40
Tipo de sistema 339, 344
tipos de defectos 32, 96, 163, 167, 340
tipos de herramientas 523, 533, 545, 549
Tipos de pruebas VI, X, 81, 147, 613
tipos de revisión 169, 174, 175
tipos de revisiones XVI, 169, 175, 177
tolerancia 93
toolsmith 398
transición de estados XVI, 257, 277, 279, 280, 284, 285, 307, 308, 309, 310, 311, 312, 313, 356, 360, 367, 373, 383
Trazabilidad 212, 232, 237

U

utilización de recursos 97, 101, 532

V

validación 90, 91, 94, 95, 145, 303, 582
valores límite XVI, 17, 257, 263, 264, 265, 270, 272, 284, 285, 286, 288, 289, 290, 294, 296, 300, 301, 303, 304, 318, 354, 360, 367, 371, 381
ventajas 225, 331, 471, 621
verificación 90, 91, 94, 95, 113, 148, 175, 564
volumen 126, 131, 223, 225, 289, 532, 543

W

WBS 416, 417, 628

Work Breakdown Structure 628

Acerca de los Autores

Rex Black: Con un cuarto de siglo de experiencia en software e ingeniería de sistemas, Rex Black es el presidente de RBCS (www.rbcus.com), un Líder en Pruebas de Software, Hardware y Sistemas. Por más de quince años, RBCS ha proveído servicios de consultoría, capacitación y tercerización para las Pruebas de Software y Hardware. Rex es el autor más productivo que hay en el campo de las Pruebas de Software hoy en día. De su primer libro famoso, *Managing the Testing Process*, se han vendido más de 50.000 ejemplares en todo el mundo, en los idiomas japonés, chino e hindú, y ahora está en su tercera edición. Así mismo se han vendido decenas de miles de sus otros libros también acerca de las Pruebas, *Advanced Software Testing: Volume I*, *Advanced Software Testing: Volume II*, *Advanced Software Testing: Volume III*, *Critical Testing Processes*, *Foundations of Software Testing* y *Pragmatic Software Testing*. Escribió más de treinta artículos, presentó cientos de artículos científicos, talleres y seminarios, y expuso en más de cincuenta conferencias y eventos en todo el mundo. Rex es el inmediato ex presidente del International Software Testing Qualifications Board y el American Software Testing Qualifications Board.

Gary Rueda Sandoval: Con 20 años de experiencia en el desarrollo y el aseguramiento de la calidad de software, Gary Rueda Sandoval es el primer instructor en Latinoamérica autorizado por la ISQB para dictar cursos en español, es el Presidente y Consultor Principal de Business Innovations S.R.L. En los últimos años ha capacitado en cursos ISTQB y Pruebas de Software a muchos profesionales en seminarios realizados en México, Costa Rica, Colombia, Ecuador, Perú, Bolivia y Chile. Miembro fundador del Comité Hispanoamericano de Calificaciones de Pruebas de Software - HASTQB. Consultor para la industria energética, Jefe de Desarrollo de Software/Pruebas y creador de herramientas de pruebas para la generación inteligente de casos y datos de prueba. Consultor en proyectos complejos para la empresa Versata (soluciones utilizadas por la NASA). Cuenta con el Certificado en ISTQB Nivel Avanzado (Test Manager), Certificado en Versata/Tenfold e IBM Rational Sales. Es Licenciado en Ingeniería Informática de la Universidad TFH Berlín, Alemania, Licenciatura en Administración de Empresas de la Universidad de Rushmore, USA.

Miles de profesionales han utilizado este material de estudio para tener éxito en su trabajo y carrera profesional con las Pruebas de Software

Fundamentos de Pruebas de Software

Las Pruebas de Software son la parte más del desarrollo y despliegue de los productos de software. Las Pruebas de Software son predominantemente vistas como una actividad periférica, casi sin importancia dentro del desarrollo del software. Un cambio de actitud y un buen programa de educación fundamentalmente las Pruebas de Software pueden reducir tiempo y costos de problemas normalmente asociados con el lanzamiento del nuevo software y minimizar el riesgo implicado.

Este libro proporcionará el conocimiento esencial para ser un profesional en Pruebas, que incluye:

- Fundamentos de Pruebas
- Pruebas a través del Ciclo de Vida de Software
- Técnicas Estáticas
- Técnicas de Diseño de Pruebas

Cabe señalar que este libro no es sólo para los probadores sino también para quienes están encargados de la adquisición de software en general, gerentes de tecnología, administradores de calidad, ingenieros de diseño, programadores, analistas de sistemas, jefes de proyectos de software, analistas, arquitectos, desarrolladores, administradores y profesores de IT.

Asimismo, el libro está diseñado para el autoestudio. El contenido comprende el programa de estudio necesario para aprobar el examen de certificación nivel básico definido por el ISTQB versión 2011 (*Syllabus 2011*).



Rex Black

Con un total de seis años de experiencia en software e Ingeniería de sistemas, Rex Black es el presidente de RBCS (www.rbcsonline.com), un Líder en Pruebas de Software, Hardware y Sistemas. Por más de siete años, RBCS ha provisto servicios de consultoría, capacitación y formación para las Pruebas de Software y Hardware. Rex es el autor de los libros "Software Testing Qualifications Board Certified Tester" y "Practical Software Testing". De acuerdo con el libro "Managing the Testing Process", se han vendido más de 50,000 ejemplares en todo el mundo, en los idiomas inglés, chino e Hindi, y ya está en su tercera edición. Adicionalmente, se han vendido decenas de miles de sus otros libros, incluyendo "Software Testing: A Practical Approach", "Advanced Software Testing: Volume I, Advanced Software Testing: Volume II, Advanced Software Testing: Volume III, Critical Testing Processes, Foundation of Software Testing y Pragmatic Software Testing". Rex es un conferencista invitado en numerosas conferencias internacionales, talleres y seminarios, y expuso en más de cincuenta conferencias y eventos en todo el mundo. Rex es el implementor ex presidente de International Software Testing Qualifications Board y el American Software Testing Qualifications Board.



Gary Rueda Sandóval

Con 20 años de experiencia en el desarrollo y el aseguramiento de la calidad de software, Gary Rueda Sandóval es el primer instructor en Latinoamérica autorizado por la ISTQB para dictar cursos en español, es el Presidente y Consultor Principal de Business Innovations S.R.L. En los últimos años ha capacitado en cursos ISTQB y Pruebas Soporte a más de 1000 profesionales de software en Argentina, Brasil, Chile, Perú, Uruguay, Paraguay, Ecuador, Bolivia y Chile. Miembro fundador del Comité Hispanoamericano de Certificaciones de Pruebas de Software - HASTQB. Consultor para la industria energética, Jefe de Desarrollo de Software/Pruebas y creador de herramientas de pruebas para la industria energética. Ha trabajado en la NASA en la creación de las pruebas para el sistema de control de la nave espacial (soluciones utilizadas por la NASA). Cuenta con el Certificado en ISTQB Nivel Avanzado (Test Manager), Certificado en Veracruz/Infonavit e IBM Rational Test. Es Licenciado en Ingeniería Informática de la Universidad ITESM, México, licenciatura en Administración de Empresas de la Universidad de Rushmore, USA.

Editorial RBCS, Inc.

Precio: 59.99 US\$

