Probador Certificado del ISTQB®

Programa de Estudio de Nivel Avanzado

Analista de Pruebas

Traducción realizada por el

Spanish Software Testing Qualifications Board

Versión ES 001.47

Basada en el Programa de Estudio

“Certified Tester - Advanced Level Syllabus, Test Analyst, Version 2012”

International Software Testing Qualifications Board

Subgrupo de Trabajo de Analista de Pruebas de Nivel Avanzado: Judy McKay (Presidente), Mike Smith, Erik Van Veenendaal; 2010-2012.

Historial de Revisiones

Versión	Fecha	Observaciones
ISEB v1.1	04SEP01	ISEB Practitioner Syllabus.
ISTQB 1.2E	SEP03	Programa de estudio de Nivel Avanzado ISTQB de EOQ-SG.
V2007	12OCT07	Programa de estudio de Probador Certificado de Nivel Avanzado, versión 2007.
D100626	26JUN10	Incorporación de cambios según aceptados en 2009, separación de capítulos para los distintos módulos.
D101227	27DIC10	Aceptación de cambios de formato y correcciones que no afectan al significado de las frases.
D2011	23OCT11	Cambio para dividir los programas de estudio, objetivos de aprendizaje reelaborados y cambios en el texto para coincidir con los objetivos de aprendizaje. Incorporación de los objetivos de aprendizaje.
Alfa 2012	09MAR12	Incorporación de todos los comentarios derivados de los NB recibidos a partir de la entrega de octubre.
Beta 2012	07ABR12	Incorporación de todos los comentarios derivados de los NB recibidos a partir de la entrega Alfa.
Beta 2012	07ABR12	Versión Beta enviada a GA.
Beta 2012	08JUN12	Versión editada entregada a NB.
Beta 2012	27JUN12	Incorporación de comentarios EWG y Glosario.
RC 2012	15AGO12	Versión candidata de entrega - ediciones NB finales incluidas.
GA 2012	19OCT12	Ediciones finales y limpieza para entrega GA.

Índice General

Agradecimientos

Este documento ha sido elaborado por un equipo principal del Subgrupo de Trabajo de Nivel Avanzado - Analista de Pruebas del International Software Testing Qualifications Board: Judy McKay (Presidenta), Mike Smith, Erik van Veenendaal.

El equipo principal agradece las sugerencias y aportaciones del equipo de revisión y de los comités nacionales.

Cuando se finalizó el programa de estudio del Nivel Avanzado, formaban parte del Grupo de Trabajo de Nivel Avanzado los siguientes miembros (por orden alfabético):

Graham Bath, Rex Black, Maria Clara Choucair, Debra Friedenberg, Bernard Homès (Vicepresidente), Paul Jorgensen, Judy McKay, Jamie Mitchell, Thomas Mueller, Klaus Olsen, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Jan Sabak, Hans Schaefer, Mike Smith (Presidente), Geoff Thompson, Erik van Veenendaal, Tsuyoshi Yumoto.

Las siguientes personas participaron en la revisión, comentarios y votación del presente programa de estudio:

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

Este documento fue hecho público formalmente por la Asamblea General del ISTQB® el 19 de octubre de 2012.

La traducción del inglés al español de la versión original ha sido revisada por Gustavo Márquez Sosa (Spanish Software Testing Qualifications Board).

0. Introducción a Este Programa de Estudio

0.1 Objetivo de este Documento

Este programa de estudio constituye la base para la Cualificación Internacional de Pruebas de Software de Nivel Avanzado para Analista de Pruebas. El ISTQB® ofrece el presente programa de estudio:

1. a los Comités Nacionales, para que lo traduzcan a su idioma local y para acreditar a los proveedores de formación. Los Comités Nacionales podrán adaptar el programa de estudio a las necesidades específicas de su idioma y modificar las referencias para adaptarlas a sus publicaciones locales.
2. a los Comités de Exámenes, para que puedan crear preguntas de examen en su idioma local que se adapten a los objetivos de aprendizaje de cada programa de estudio.
3. a los proveedores de formación, para que elaboren los materiales didácticos y determinen las metodologías de enseñanza apropiadas.
4. a los candidatos a la certificación, para que se preparen para el examen (como parte de un curso de formación o de manera independiente).
5. a la comunidad internacional de ingeniería de sistemas y software, para que la actividad de pruebas de software y sistemas avance, y como referencia para la elaboración de libros y artículos.

El ISTQB® podrá autorizar que otras entidades utilicen este programa de estudio con otros fines, siempre y cuando soliciten y obtengan la correspondiente autorización previa por escrito.

0.2 Generalidades

El Nivel Avanzado consta de tres programas de estudio independientes.

- Jefe de Pruebas.
- Analista de Pruebas.
- Analista de Pruebas Técnicas.

El documento Descripción del Nivel Avanzado (“Advanced Level Syllabus (2012) Overview”) [ISTQB_AL_OVIEW] incluye la siguiente información:

- Resultados de negocio de cada programa de estudio.
- Resumen de cada programa de estudio.
- Relaciones entre los programas de estudio.
- Descripción de los niveles de conocimiento (niveles K).
- Anexos.

0.3 Objetivos de Aprendizaje Objeto de Examen

Los objetivos de aprendizaje respaldan los resultados de negocio y se utilizan para elaborar el examen para lograr la Certificación de Analista de Pruebas de Nivel Avanzado. Por lo general, todas las partes de este programa de estudio pueden ser objeto de examen en el nivel K1. Es decir, el candidato reconocerá, memorizará y recordará un término o concepto. Los objetivos de aprendizaje correspondientes a los niveles K2, K3 y K4 se muestran al principio de cada capítulo.

1. Proceso de Pruebas - 300 minutos

Palabras Clave

caso de prueba concreto, criterios de salida, caso de prueba de alto nivel, caso de prueba lógico, caso de prueba de bajo nivel, control de la prueba, diseño de la prueba, ejecución de la prueba, implementación de la prueba, planificación de la prueba

Objetivos de Aprendizaje para el Proceso de Pruebas

1.2 Pruebas en el Ciclo de Vida de Desarrollo de Software

AP[1]-1.2.1 (K2) Explicar cómo y por qué varían los tiempos y el nivel de implicación de los Analistas de Pruebas cuando trabajan con distintos modelos de ciclo de vida.

1.3 Monitorización, Planificación y Control de Pruebas

AP-1.3.1 (K2) Resumir las actividades llevadas a cabo por el Analista de Pruebas para contribuir a la planificación y al control de las pruebas.

1.4 Análisis de Pruebas

AP-1.4.1 (K4) Analizar un escenario dado, incluyendo una descripción de proyecto y un modelo de ciclo de vida, para establecer las tareas adecuadas del Analista de Pruebas durante las fases de análisis y diseño.

1.5 Diseño de Pruebas

AP-1.5.1 (K2) Explicar por qué los implicados deben entender las condiciones de prueba.

AP-1.5.2 (K4) Analizar un escenario de proyecto para establecer el uso más apropiado de los casos de prueba de bajo nivel (concretos) o de alto nivel (lógicos).

1.6 Implementación de pruebas

AP-1.6.1 (K2) Describir los criterios de salida típicos para el análisis de pruebas y del diseño de pruebas y explicar cómo el cumplimiento de dichos criterios afecta al esfuerzo de implementación de las pruebas.

1.7 Ejecución de Pruebas

AP-1.7.1 (K3) En un escenario dado, establecer los pasos y las consideraciones que deberían adoptarse a la hora de ejecutar las pruebas.

1.8 Evaluación de los Criterios de Salida y Gestión de Información de la Prueba

AP-1.8.1 (K2) Explicar por qué es importante la información sobre el estado de ejecución de los casos de prueba.

1.9 Actividades de Cierre de Pruebas

AP-1.9.1 (K2) Poner ejemplos de productos de trabajo que el Analista de Pruebas debería entregar durante las actividades de cierre de pruebas.

[1] AP es el acrónimo de **A**nalista de **P**ruebas.

1.1 Introducción

En el Programa de Estudio de Nivel Fundamentos ISTQB®, se describe el proceso de pruebas básico en el que se incluyen las siguientes actividades:

- Planificación, monitorización y control.
- Análisis y diseño.
- Implementación y ejecución.
- Evaluación de los criterios de salida y gestión de información de la prueba[2].
- Actividades de cierre de pruebas.

En el Nivel Avanzado, algunas de estas actividades se abordan de manera independiente para ofrecer un mayor perfeccionamiento y optimización de los procesos, para ajustarse mejor al ciclo de vida de desarrollo de software, y para facilitar la correcta monitorización y el control efectivo de las pruebas Las actividades en este nivel se consideran como sigue:

- Planificación, monitorización y control.
- Análisis.
- Diseño.
- Implementación.
- Ejecución.
- Evaluación de los criterios de salida y gestión de información de la prueba.
- Actividades de cierre de pruebas.

Estas actividades pueden llevarse a cabo de manera secuencial, si bien algunas de ellas pueden llevarse a cabo en paralelo. Así por ejemplo, el diseño podría llevarse a cabo en paralelo a la implementación (por ejemplo, pruebas exploratorias). Establecer las pruebas y los casos de prueba adecuados, diseñarlos y ejecutarlos constituyen las principales áreas de concentración de los Analistas de Pruebas. Si bien es importante conocer los demás pasos del proceso de prueba, la mayoría del trabajo de los Analistas de Pruebas generalmente tiene lugar durante las actividades de análisis, diseño, implementación y ejecución del proyecto de prueba.

Los probadores de nivel avanzado se enfrentan a múltiples retos a la hora de introducir los diferentes aspectos de pruebas descritos en este programa de estudio dentro del contexto de sus propias organizaciones, equipos y tareas. Es importante tener en cuenta los distintos ciclos de vida de desarrollo de software, así como el tipo de sistema que se está probando, ya que estos factores pueden influir en el enfoque de prueba.

1.2 Pruebas en el Ciclo de Vida de Desarrollo de Software

El enfoque de pruebas de ciclo de vida a largo plazo debe estudiarse y definirse como parte de la estrategia de prueba. El momento de intervención del Analista de Pruebas varía según los distintos ciclos de vida, asimismo el grado de implicación, el tiempo necesario, la información disponible y las expectativas también pueden variar mucho. Dado que los procesos de prueba no se producen de una forma aislada, el Analista de Pruebas debe ser consciente de los momentos en los que puede facilitarse información a las demás áreas relacionadas de la organización, tales como:

- Ingeniería y gestión de requisitos - revisiones de requisitos.
- Gestión de proyectos - establecimiento del calendario de entradas.
- Gestión de la configuración y del cambio - construcción de pruebas de verificación, control de versiones.
- Desarrollo de software - anticipar qué va a ocurrir, y cuándo.
- Mantenimiento de software - gestión de defectos, tiempo de respuesta (es decir, el tiempo transcurrido desde el descubrimiento del defecto hasta su resolución).
- Soporte técnico - documentación exacta sobre soluciones temporales.
- Elaboración de documentación técnica (por ejemplo, especificaciones de diseño de bases de datos) - datos de entrada para estos documentos, así como la revisión técnica de los mismos.

[2] Antes “elaboración de informes”.

Las actividades de prueba deben estar alineadas con el modelo de ciclo de vida de desarrollo de software elegido, cuyo carácter puede ser secuencial, iterativo o incremental. Por ejemplo, en el modelo-V secuencial, el proceso de pruebas fundamental ISTQB® aplicado al nivel de pruebas de sistema podría alinearse como sigue:

- La planificación de la prueba de sistema concurre con la planificación del proyecto, y el control de prueba persiste hasta la finalización de la ejecución y el cierre de las pruebas de sistema.
- El análisis y diseño de las pruebas de sistema tienen lugar de forma concurrente con la especificación de requisitos, la especificación de diseño del sistema, la especificación de diseño de la arquitectura (de alto nivel), y la especificación de diseño (de bajo nivel) de componentes.
- La implementación del entorno de pruebas de sistema (por ejemplo, camas de prueba, aparejos de prueba) puede iniciarse durante la fase de diseño del sistema, pero normalmente el grueso concurre con la codificación y las pruebas de componente, a menudo, el trabajo asociado a las tareas de implementación de las pruebas de sistema se extiende hasta pocos días antes del inicio de la ejecución de las pruebas.
- Las pruebas de sistema se ejecutan cuando se cumplen todos los criterios de entrada de las pruebas de sistema (o cuando éstos no son exigibles), lo que normalmente supone que, como mínimo, las pruebas de componente, y a veces también las pruebas de integración de componentes, han concluido. La ejecución de pruebas de sistema continúa hasta que se cumplan los criterios de salida de las pruebas de sistema.
- La evaluación de los criterios de salida de las pruebas de sistema y la gestión de información sobre los resultados de las pruebas de sistema se realizarán durante el tiempo de ejecución de las pruebas de sistema, generalmente con mayor frecuencia y urgencia a medida que se van aproximando los plazos del proyecto.
- Las actividades de cierre de las pruebas de sistema se ejecutan cuando se cumplen los criterios de salida de las pruebas de sistema y una vez finalizadas las pruebas de sistema, si bien en ocasiones pueden retrasarse hasta después de finalizar las pruebas de aceptación y todas las actividades del proyecto.

Los modelos iterativos e incrementales pueden no seguir el mismo orden de tareas y pueden excluir algunas tareas. Así por ejemplo, un modelo iterativo puede utilizar un conjunto reducido de procesos de pruebas estándar para cada iteración. Para cada iteración se pueden realizar las tareas de análisis y diseño, implementación y ejecución, así como evaluación y gestión de la información, mientras que la planificación solo se realiza al inicio del proyecto y la elaboración de informes de cierre al final. En los proyectos Ágiles, a menudo, se aplica un proceso menos formal y se establece una relación de colaboración mucho más estrecha que permite implementar cambios de una forma más sencilla dentro del proyecto. Dado que la metodología ágil constituye un proceso “ligero”, conlleva documentación de pruebas menos detalladas al disponer de métodos de comunicación más rápidos, tales como reuniones diarias “de pie[3]” (llamadas de “de pie” porque son muy rápidas, normalmente de entre 10 y 15 minutos, para que nadie necesite sentarse y todos mantengan una participación activa).

Los proyectos Ágiles, de entre todos los modelos de ciclo de vida, son los que requieren la intervención más temprana del Analista de Pruebas. El Analista de Pruebas debería esperar verse involucrado desde el inicio del proyecto, trabajando en colaboración con los desarrolladores a medida que éstos realizan su trabajo inicial de arquitectura y diseño. Las revisiones pueden no ser formales, pero son constantes a medida que el software evoluciona. Se espera que el Analista de Pruebas intervenga durante todo el proyecto y esté disponible para el equipo. Debido a esta inmersión, los miembros de equipos Ágiles normalmente se dedican a un único proyecto y están plenamente involucrados en todos los aspectos del mismo.

Los modelos iterativos/incrementales van desde un enfoque Ágil, donde cabe esperar que se produzcan cambios a medida que el software evoluciona, hasta modelos de desarrollo iterativos/incrementales, que existen dentro de un modelo-V (a menudo denominado modelo iterativo embebido). En el caso de un modelo iterativo embebido, lo habitual es que el Analista de Pruebas participe en los aspectos estándar de planificación y diseño, para a continuación pasar a desempeñar un papel más interactivo a medida que el software se desarrolla, se prueba y se despliega.

[3] “Reunión diaria de pie” es la traducción de “daily stand up meeting”.

Independientemente del ciclo de vida de desarrollo de software que se utilice, es importante que el Analista de Pruebas conozca qué se espera de su intervención, así como el momento adecuado. Se utilizan muchos modelos híbridos, tales como el iterativo dentro de un modelo-V descrito anteriormente. En este sentido, a menudo, es el Analista de Pruebas el que tiene que establecer cuál es el papel más efectivo y el trabajo asociado al mismo, en lugar de depender de la definición de un modelo establecido que indique el momento en que debe involucrarse.

1.3 Planificación, Monitorización y Control de la Prueba

Esta sección se centra en los procesos de planificación, monitorización y control de las pruebas.

1.3.1 Planificación de la Prueba

La planificación de la prueba tiene lugar principalmente al inicio del esfuerzo de prueba e implica la identificación y la planificación de todas las actividades y los recursos necesarios para cumplir la misión y los objetivos identificados en la estrategia de prueba. Durante la planificación de la prueba es importante que el Analista de Pruebas, en colaboración con el Jefe de Pruebas, tenga en cuenta y planifique los siguientes aspectos:

- Asegurarse de que los planes de prueba no se limitan a pruebas funcionales. El plan de pruebas debe tener en cuenta todos los tipos de pruebas y programarlos en consecuencia. Así por ejemplo, además de las pruebas funcionales, el Analista de Pruebas puede ser responsable de las pruebas de usabilidad. Ese tipo de prueba también debe abordarse en un documento de plan de prueba.
- Repasar las estimaciones de pruebas con el Jefe de Pruebas y garantizar que se reserva tiempo suficiente para la adquisición y validación del entorno de pruebas.
- Planificar pruebas de configuración. Si se pueden combinar múltiples tipos de procesadores, sistemas operativos, máquinas virtuales, navegadores o periféricos en diferentes configuraciones posibles, prever el uso de técnicas de prueba que ofrezcan una cobertura adecuada para estas combinaciones.
- Planificar pruebas de documentación. A los usuarios se les provee el software y documentación. Para ser efectiva, la documentación debe ser exacta. El Analista de Pruebas debe reservar tiempo para comprobar la documentación y es posible que tenga que trabajar con los técnicos responsables de la documentación para conseguir elaborar los datos que se utilizarán en capturas de pantalla y videoclips.
- Planificar pruebas de procedimientos de instalación. Los procedimientos de instalación, al igual que los procedimientos de copias de seguridad y restauración, deben ser suficientemente probados. Estos procedimientos pueden ser más críticos que el propio software, si el software no se puede instalar, no servirá para nada. Eso puede resultar difícil de prever, ya que a menudo el Analista de Pruebas es el que realiza las pruebas iniciales de un sistema que ha sido preconfigurado sin disponer de los procesos finales de instalación.
- Planificar las pruebas para su alineación con el ciclo de vida de software. La ejecución secuencial de tareas no encaja en la mayoría de los calendarios. A menudo, es necesario realizar varias tareas de forma simultánea. El Analista de Pruebas debe conocer el ciclo de vida seleccionado y ser consciente de lo que se espera de su intervención durante las fases de diseño, desarrollo e implementación del software. Esto también incluye reservar tiempo para las pruebas de confirmación y regresión.
- Dejar tiempo suficiente para la identificación y el análisis de riesgos con un equipo transversal. Si bien normalmente la organización de las sesiones de gestión de riesgos no entra dentro de sus responsabilidades, el Analista de Pruebas debería participar activamente en estas actividades.

Es posible que existan relaciones complejas entre la base de prueba, las condiciones de prueba, y los casos de prueba, por lo que es posible que entre estos productos de trabajo existan relaciones de tipo “muchos a muchos[4]”. Para que la planificación y el control de la prueba sean efectivos, es necesario conocer estas relaciones. El Analista de Pruebas suele ser la persona ideal para determinar estas relaciones y trabajar para separar las dependencias al máximo.

[4] Relación de “n a m”.

1.3.2 Monitorización y Control de la Prueba

Si bien la monitorización y el control de la prueba, normalmente, competen al Jefe de Pruebas, el Analista de Pruebas es el que aporta las mediciones que hacen posible este control.

A lo largo del ciclo de vida de desarrollo de software, es necesario recopilar una serie de datos cuantitativos (por ejemplo, el porcentaje de actividades de la planificación finalizadas, el porcentaje de cobertura alcanzado, el número de casos de prueba pasados y fallados). En cada caso, debe definirse una línea base (es decir, un estándar de referencia) y posteriormente debe hacerse un seguimiento del progreso en relación con dicha línea base. Si bien el Jefe de Pruebas es responsable de recopilar y comunicar la información métrica resumida, el Analista de Pruebas es el que recopila la información asociada a cada métrica. Cada caso de prueba que se realice, cada informe de defecto que se elabore, cada hito que se alcance se reflejará en las métricas generales del proyecto. Es importante que la información que se introduzca en las distintas herramientas de seguimiento sea lo más exacta posible para que las métricas reflejen la realidad.

Unas métricas exactas permitirán a los mandos correspondientes gestionar el proyecto (monitorizar) y poner en marcha los cambios que sean necesarios (controlar). Así por ejemplo, el hecho de que un área del software reporte un alto número de defectos puede indicar que dicha área requiere un esfuerzo de prueba adicional. La información sobre requisitos y cobertura de riesgos (trazabilidad) puede utilizarse para priorizar el trabajo pendiente y asignar recursos. La información sobre la causa raíz se emplea para determinar cuáles son las áreas de mejora del proceso. Si los datos registrados son exactos, el proyecto puede controlarse y los implicados pueden recibir información precisa sobre su estado. Si la planificación tiene en cuenta los datos recopilados en proyectos pasados, podrán planificarse proyectos futuros de una forma más efectiva. Los datos exactos pueden utilizarse de mil maneras distintas. Es competencia del Analista de Pruebas asegurar que los datos son exactos, oportunos y objetivos.

1.4 Análisis de la Prueba

Durante la planificación de la prueba, se establece el alcance del proyecto de prueba. El Analista de Pruebas se sirve de esta definición del alcance para:

- Analizar la base de prueba.
- Identificar las condiciones de prueba.

Para que el Analista de Pruebas pueda proceder de manera efectiva al análisis de las pruebas, deben cumplirse los siguientes criterios de entrada:

- Hay un documento que describe el objeto de prueba que puede servir como base de prueba.
- Este documento ha sido objeto de revisión con resultados razonables y se ha actualizado en consecuencia tras la misma.
- Se dispone de un presupuesto y de un calendario razonables para llevar a cabo el trabajo de prueba pendiente para este objeto de prueba.

Normalmente las condiciones de prueba se identifican mediante el análisis de la base de prueba y los objetivos de prueba. En algunas situaciones, cuando la documentación es obsoleta o no existente, las condiciones de prueba pueden establecerse mediante intercambios con los implicados relevantes (por ejemplo, en talleres o durante la planificación de las iteraciones[5]). A continuación estas condiciones se utilizan para establecer qué debe probarse, aplicando las técnicas de diseño de prueba previstas en la estrategia de pruebas y/o en el plan de prueba.

Si bien las condiciones de prueba suelen ser específicas del elemento objeto de prueba, el Analista de Pruebas debe tener en cuenta ciertas consideraciones estándar.

- Por lo general, se recomienda definir las condiciones de prueba en distintos niveles de detalle. Inicialmente, se definen condiciones de alto nivel para identificar los objetivos generales de las pruebas, tales como “funcionalidad de la pantalla X”. A continuación, se definen condiciones más detalladas, como base de casos de prueba específicos, tales como “la pantalla X rechaza

[5] “sprints”

un número de cuenta con un dígito menos de lo previsto en la longitud correcta". El uso de este tipo de enfoque jerárquico para definir las condiciones de prueba puede ayudar a garantizar que los elementos de alto nivel tengan una cobertura suficiente.
- Si se han definido los riesgos de producto, entonces deben identificarse las condiciones de prueba necesarias para abordar cada riesgo de producto y asociarlas a dicho elemento de riesgo.

Una vez concluidas las actividades de análisis de las pruebas, el Analista de Pruebas debe saber qué pruebas específicas se deben diseñar para satisfacer las necesidades de las áreas asignadas del proyecto de prueba.

1.5 Diseño de la Prueba

Cumpliendo con el alcance establecido durante la planificación de la prueba, el proceso de prueba avanza a medida que el Analista de Pruebas diseña las pruebas que se implementarán y ejecutarán. El proceso del diseño de pruebas incluye las siguientes actividades:
- Establecer en qué áreas de prueba son más adecuados casos de prueba de bajo nivel (concretos) o de alto nivel (lógicos).
- Determinar las técnicas de diseño de casos de prueba que aporten la cobertura de prueba necesaria.
- Crear casos de prueba que practiquen las condiciones de prueba identificadas.

A lo largo de todo este proceso deben aplicarse los criterios de priorización identificados durante el análisis de riesgos y la planificación de la prueba, desde el análisis y diseño hasta la implementación y ejecución.

Dependiendo de los tipos de pruebas que se estén diseñando, uno de los criterios de entrada para el diseño de pruebas puede ser la disponibilidad de herramientas durante el trabajo de diseño.

Durante el diseño de pruebas, es importante recordar lo siguiente:
- Algunos elementos de prueba se abordan mejor definiendo solo las condiciones de prueba en lugar de avanzar en la definición de pruebas programadas. En este caso, las condiciones de prueba definidas deberían utilizarse como guía para las pruebas no programadas.
- Los criterios de paso/fallo deben estar claramente identificados.
- Las pruebas deben diseñarse para que otros probadores puedan entenderlas, no solo el autor. Si el autor no es la persona que va a ejecutar la prueba, otros probadores tendrán que leer y conocer las pruebas previamente especificadas para identificar los objetivos de la prueba y la importancia relativa de la prueba.
- Las pruebas también deben ser comprensibles para otros implicados, tales como los desarrolladores que revisarán las pruebas y los auditores que tendrán que aprobarlas.
- Las pruebas deben diseñarse para cubrir todas las interacciones del software con los actores (por ejemplo, usuarios finales, otros sistemas), y no únicamente las interacciones que se produzcan en la interfaz de usuario visible. Las comunicaciones entre procesos, la ejecución por lotes y otras interrupciones también interactúan con el software y pueden contener defectos, por lo que el Analista de Pruebas debe diseñar pruebas para mitigar estos riesgos.
- Se deberían diseñar pruebas para probar las interfaces entre los distintos objetos de prueba.

1.5.1 Casos de Prueba Concretos y Lógicos

Una de las tareas del Analista de Pruebas es establecer los mejores tipos de casos de prueba para una situación dada. Los casos de prueba concretos ofrecen toda la información y los procedimientos específicos que el probador necesita para ejecutar el caso de prueba (incluyendo cualquier requisito de datos) y verificar los resultados. Los casos de prueba concretos son útiles cuando los requisitos están bien definidos, cuando el personal de pruebas es más inexperto y cuando se requiere una verificación externa de las pruebas, tales como auditorías. Los casos de prueba concretos ofrecen una excelente reproducibilidad (es decir, otro probador obtendrá los mismos resultados), pero también pueden requerir un esfuerzo de mantenimiento importante y tienden a limitar la inventiva del probador durante la fase de ejecución.

Los casos de prueba lógicos ofrecen pautas sobre qué debería probarse, pero permiten al Analista de Pruebas modificar los datos reales o incluso el procedimiento seguido durante la ejecución de la prueba. Los casos de prueba lógicos pueden ofrecer una mejor cobertura que los casos de prueba concretos porque variarán ligeramente cada vez que se ejecuten. Esto también supone una pérdida de reproducibilidad. Los casos de prueba lógicos son idóneos cuándo los requisitos no están bien definidos, cuando el Analista de Pruebas que va a ejecutar la prueba tiene experiencia tanto en pruebas como en el producto, y cuando no se requiere documentación formal (por ejemplo, no van a realizarse auditorías). Los casos de prueba lógicos pueden definirse en una etapa temprana del proceso de requisitos, cuando los requisitos aún no están bien definidos. Estos casos de prueba pueden utilizarse para desarrollar casos de prueba concretos cuando los requisitos pasan a estar más definidos y a ser más estables. En este caso, la creación del caso de prueba se lleva a cabo de una forma secuencial, pasando de lógico a concreto solo con los casos de prueba concretos que se van a utilizar en la ejecución.

1.5.2 Creación de Casos de Prueba

Los casos de prueba se diseñan mediante la elaboración y refinamiento “paso a paso” de las condiciones de prueba identificadas, utilizando las técnicas de prueba (véase el Capítulo 3) identificadas en la estrategia de prueba y/o el plan de prueba. Los casos de prueba deben ser repetibles, verificables y trazables con la base de prueba (por ejemplo, requisitos) según lo estipulado en la estrategia de pruebas aplicada.

El diseño de los casos de prueba incluye la identificación de lo siguiente:

- Objetivo.
- Precondiciones, tales como los requisitos del proyecto o de entorno de prueba localizado y los planes para su entrega, el estado del sistema, etc.
- Requisitos de los datos de prueba (tanto datos de entrada para el caso de prueba como datos que deben existir en el sistema para ejecutar el caso de prueba).
- Resultados esperados.
- Postcondiciones, tales como datos afectados, estado del sistema, disparadores[6] para procesamientos subsiguientes, etc.

Antes de crear los casos de prueba se debe establecer su nivel de detalle, el cual afecta tanto al coste de desarrollo como al nivel de repetibilidad durante la ejecución. Los casos de prueba menos detallados permiten una mayor flexibilidad al Analista de Pruebas a la hora de ejecutar el caso de prueba y ofrecen una oportunidad para investigar áreas que sean potencialmente interesantes. Un menor nivel de detalle, no obstante, también conlleva una menor reproducibilidad.

A menudo, la definición del resultado esperado de una prueba plantea un reto particular. Calcular el resultado esperado de forma manual puede resultar tedioso y propenso a errores, si es posible, es preferible encontrar o crear un oráculo de prueba automatizado. Al identificar el resultado esperado, los probadores no sólo deben tener en cuenta las salidas en la pantalla, sino también las postcondiciones de los datos y del entorno. Si la base de prueba está claramente definida, identificar el resultado correcto, en teoría, debería ser sencillo. Sin embargo, las bases de prueba suelen ser vagas, contradictorias, con poca cobertura de áreas clave o totalmente nulas. En estos casos, el Analista de Pruebas debe disponer de, o tener acceso a, conocimiento experto en la materia. Asimismo, incluso en aquellos casos en los que la base de prueba está bien especificada, las interacciones complejas de estímulos y respuestas complejas pueden dificultar la definición de los resultados esperados, por lo que un oráculo de pruebas resulta esencial. Ejecutar un caso de prueba sin disponer de una forma para establecer la corrección de los resultados supone pocas ventajas o valor añadido, y en muchas ocasiones genera informes de fallo falsos o genera una falsa confianza en el sistema.

Las actividades descritas anteriormente pueden aplicarse a todos los niveles de prueba, si bien la base de prueba variará. Así por ejemplo, las pruebas de aceptación de usuario pueden basarse principalmente en la especificación de requisitos, casos de uso y procesos de negocio definidos, mientras que las pruebas de componente pueden basarse principalmente en las especificaciones de

[6] “triggers”

diseño de bajo nivel, las historias de usuario y el propio código. Es importante recordar que estas actividades se realizan durante todos los niveles de pruebas, si bien el objetivo de la prueba puede variar. Así por ejemplo, las pruebas funcionales a nivel de unidad tienen por objeto garantizar que un componente en particular ofrece la funcionalidad prevista en el diseño detallado de dicho componente. Las pruebas funcionales a nivel de integración verifican que los componentes interactúan juntos y ofrecen funcionalidad a través de su interacción. A nivel de sistema, la funcionalidad extrema a extremo debería ser un objetivo de las pruebas. Cuándo se analizan y diseñan pruebas, es importante recordar tanto el nivel objetivo de la prueba como el objetivo de la prueba. Esto permitirá determinar el nivel de detalle requerido, además de las herramientas necesarias (por ejemplo, controladores y stubs a nivel de pruebas de componente).

Durante el desarrollo de las condiciones de prueba y los casos de prueba, normalmente se genera cierta cantidad de documentación que da lugar a productos de trabajo. En la práctica, la medida en la que los productos de trabajo están documentados puede variar considerablemente Esto puede ser consecuencia de cualquiera de los siguientes:

- Los riesgos del proyecto (qué debe/no debe ser documentado).
- El "valor añadido" que la documentación aporta al proyecto.
- Estándares a seguir y/o la normativa a cumplir.
- El modelo de ciclo de vida empleado (por ejemplo, un enfoque Ágil aspira a generar la documentación estrictamente necesaria).
- El requisito de trazabilidad a partir de la base de prueba, mediante el análisis y el diseño de las pruebas.

En función del alcance de las pruebas, las tareas de análisis y diseño de las pruebas abordan las características de calidad del objeto de prueba. La norma ISO 25000 [ISO25000] (que sustituye a la norma ISO 9126) constituye una referencia útil. En el caso de las pruebas de hardware/sistemas software, pueden aplicarse características adicionales.

Los procesos de análisis y diseño de las pruebas pueden ampliarse entrelazándolos con revisiones y análisis estático. De hecho, el análisis de pruebas y el diseño de pruebas, a menudo, constituyen una especie de pruebas estáticas, ya que durante el proceso pueden detectarse problemas en los documentos base. Analizar y diseñar las pruebas tomando como base la especificación de requisitos es una manera excelente de preparar para una reunión de revisión de requisitos. La lectura de los requisitos para su uso en la creación de pruebas requiere conocer los requisitos y ser capaz de establecer la forma de evaluar su cumplimiento. Esta actividad, a menudo, permite identificar requisitos que no están claros, que no son susceptibles de ser probados[7] o que no tienen criterios de aceptación definidos. De manera similar, los productos de trabajo de pruebas, tales como los casos de prueba, los análisis de riesgos y los planes de prueba, deben ser objeto de revisiones.

Es posible que algunos proyectos, tales como los que siguen un ciclo de vida Ágil, solo dispongan de requisitos mínimamente documentados. A veces estos requisitos adoptan la forma de "historias de usuario" que describen pequeñas partes de la funcionalidad pero demostrables. Una historia de usuario debe incluir una definición de los criterios de aceptación. Si el software es capaz de demostrar que ha cumplido los criterios de aceptación, normalmente se considerará que está listo para su integración con las demás funcionalidades finalizadas, aunque también puede haberse integrado ya para demostrar su funcionalidad.

Durante el diseño de las pruebas, podrán definirse en detalle los requisitos de infraestructura de las pruebas necesarios, a pesar de que en la práctica estos no se completarán hasta la implementación de las pruebas. Cabe recordar que la infraestructura de las pruebas incluye más cosas además de los objetos de prueba y los productos de soporte de pruebas. Así por ejemplo, los requisitos de infraestructura pueden incluir salas, equipo, personal, software, herramientas, periféricos, equipos de comunicación, autorizaciones de usuario y demás elementos necesarios para ejecutar las pruebas.

Los criterios de salida del análisis de pruebas y del diseño de pruebas variarán en función de los parámetros del proyecto, no obstante, a la hora de definir los criterios de salida debe considerarse la

[7] Antes "requisito testeable".

inclusión de todos los elementos abordados en estas dos secciones. Es importante que los criterios sean medibles y asegurarse de que se ha facilitado toda la información y preparación necesarias para los pasos posteriores.

1.6 Implementación de las Pruebas

La implementación de las pruebas consiste en llevar a cabo el diseño de las pruebas. Esto incluye crear pruebas automatizadas, organizar las pruebas (tanto manuales como automatizadas) por su orden de ejecución, completar los datos de prueba y los entornos de pruebas, y establecer un calendario para la ejecución de las pruebas, incluyendo la asignación de recursos, para poder iniciar la ejecución del caso de prueba. Esto incluye comprobar los criterios de entrada explícitos e implícitos del nivel de prueba correspondiente y garantizar que se han cumplido los criterios de salida en los pasos anteriores del proceso. Si se han obviado los criterios de salida, bien en el nivel de prueba o en un paso del proceso de pruebas, es probable que el esfuerzo de implementación se vea afectado por retrasos en los plazos, una calidad insuficiente y un esfuerzo extraordinario imprevisto. Es importante asegurar que se han cumplido todos los criterios de salida antes de iniciar el esfuerzo de implementación de las pruebas.

A la hora de determinar el orden de ejecución, deben tenerse en cuenta muchos factores. En algunos casos, puede tener sentido organizar los casos de prueba en juegos de pruebas (es decir, grupos de casos de prueba). Esto permitirá organizar las pruebas de modo que los casos de prueba relacionados se ejecuten juntos. Si se utiliza una estrategia de pruebas basada en riesgos, el orden de prioridad del riesgo puede imponer el orden de ejecución de los casos de prueba. Puede haber otros factores que determinen el orden, tales como la disponibilidad de los recursos adecuados, el equipo, los datos y la funcionalidad a probar. No es raro que el código se entregue en secciones y que el esfuerzo de prueba tenga que coordinarse según el orden en el que el software pasa a estar disponible para su prueba. Especialmente en los modelos de ciclo de vida incrementales, es importante que el Analista de Pruebas se coordine con el equipo de desarrollo para garantizar que el software llegue a las pruebas en un orden que sea susceptible de ser probado. Durante la implementación de las pruebas, los Analista de Pruebas deben ultimar y confirmar el orden en el que deben ejecutarse las pruebas manuales y automatizadas, comprobando con especial cuidado aquellas restricciones que puedan requerir a que las pruebas se ejecuten en un orden específico. Se deben documentar y comprobar las dependencias.

El nivel de detalle y la complejidad asociada al trabajo realizado durante la implementación de las pruebas puede verse afectado por el detalle de los casos de prueba y las condiciones de prueba. En algunos casos, se debe cumplir con una normativa y las pruebas deben aportar la evidencia del cumplimiento de los estándares aplicables, tales como la DO-178B/ED 12B [RTCA DO-178B/ED-12B] de la Administración de Aviación Federal de los Estados Unidos.

Según lo especificado anteriormente, los datos de prueba son necesarios para las pruebas, y en algunos casos estos conjuntos de datos pueden ser bastante voluminosos. Durante la implementación, los Analistas de Pruebas crean datos de entrada y de entorno para cargar en bases de datos y otros repositorios. Los Analistas de Pruebas también crean los datos que se utilizarán en las pruebas de automatización guiadas por datos así como en pruebas manuales.

La implementación de las pruebas también está relacionada con el entorno o los entornos de prueba. En esta etapa, el entorno o los entornos deben configurarse y verificarse íntegramente antes de proceder a ejecutar las pruebas. Es fundamental disponer de un entorno de prueba "idóneo[8]", es decir, el entorno de pruebas debe ser capaz de permitir la exposición de los defectos presentes durante la ejecución de pruebas controladas, operar normalmente cuando no haya fallos y replicar de forma adecuada, si fuera necesario, un entorno de producción o de usuario final para niveles de prueba superiores. Es posible que, durante la ejecución de las pruebas, el entorno de prueba sea objeto de cambios en función de la aparición de cambios imprevistos, de los resultados de las pruebas o de otras consideraciones. Si durante la ejecución se producen cambios en el entorno, es importante evaluar su impacto en las pruebas que ya se han ejecutado.

[8] "adecuado al uso"

Durante la implementación de las pruebas, los probadores deben garantizar que los responsables de la creación y del mantenimiento del entorno de prueba son conocidos y están disponibles, y que todos los productos de soporte de pruebas y las herramientas de soporte y procesos asociados están listos para su uso. Esto incluye la gestión de la configuración, la gestión de defectos así como el registro y la gestión de las pruebas. Además, los Analistas de Pruebas deben verificar los procedimientos que recopilan datos para la evaluación de los criterios de salida y gestión de la información de los resultados de las pruebas.

Se recomienda adoptar un enfoque equilibrado para la implementación de las pruebas según lo establecido durante la etapa de planificación de la prueba. Así por ejemplo, las estrategias de prueba analíticas basadas en riesgos a menudo se combinan con estrategias de prueba dinámicas. En este caso, hay un porcentaje del esfuerzo de implementación de la prueba que se asigna a aquellas pruebas que no siguen guion predeterminados (no programadas, que no son pruebas mediante guion).

Las pruebas sin guion[9] no deben ser ad hoc ni carecer de objetivos ya que pueden ser impredecibles en cuanto a su duración y cobertura, a menos que estén limitadas temporal y contractualmente en virtud de un contrato[10]. A lo largo de los años, los probadores han desarrollado una variedad de técnicas basadas en la experiencia, tales como ataques, predicción de errores [Myers79] y pruebas exploratorias. El análisis de las pruebas, el diseño de las pruebas y la implementación de las pruebas siguen llevándose a cabo pero, sobre todo, durante la ejecución de las pruebas.

Cuando se sigue este tipo de estrategias de prueba dinámicas, los resultados de cada prueba afectan al análisis, al diseño y a la implementación de las pruebas posteriores. Si bien estas estrategias son ligeras y a menudo efectivas a la hora de detectar defectos, también tienen algunos inconvenientes. Estas técnicas requieren de un Analista de Pruebas experimentado, su duración puede ser difícil de predecir, la cobertura puede ser difícil de monitorizar y se puede llegar a perder la repetibilidad si no se dispone de una buena documentación o del apoyo de herramientas.

1.7 Ejecución de Pruebas

La ejecución de pruebas comienza una vez entregado el objeto de prueba y habiéndose cumplido (o no habiéndose aplicado) los criterios de entrada de la ejecución de las pruebas. Las pruebas deben ejecutarse según el plan determinado durante la implementación de las pruebas, sin embargo el Analista de Pruebas debe disponer de tiempo suficiente para asegurar la cobertura de otros escenarios de prueba y comportamientos interesantes observados durante las pruebas (cualquier fallo detectado durante estas desviaciones debe incluir una descripción de las variaciones respecto del caso de prueba mediante guion[11] que deben darse para reproducir el fallo). Esta integración de técnicas de prueba con y sin guion (por ejemplo, exploratorias) ayuda en la protección contra las pruebas fugadas[12] provocadas por vacíos en la cobertura mediante guion y eludir la paradoja del pesticida.

En el centro de la actividad de ejecución de pruebas se encuentra la comparación de los resultados reales con los resultados esperados. Los Analistas de Pruebas deben atraer la atención y concentrarse en estas tareas, de lo contrario todo el trabajo de diseño e implementación de pruebas puede ser desaprovechado si se pasan por alto fallos (resultado de falso-negativo) o se corrigen comportamientos erróneamente clasificados como incorrectos (resultados de falso positivo). Si los resultados esperados no coinciden con los resultados reales quiere decir que se ha producido una incidencia. Las incidencias deben examinarse en detalle para determinar la causa (que puede ser o no un defecto en el objeto de prueba) y recopilar datos para ayudar a su resolución (remítase al Capítulo 6 para más detalles sobre la gestión de defectos).

Cuando se detecta un fallo, la documentación de la prueba (especificación de prueba, caso de prueba, etc.) debe evaluarse detenidamente para comprobar que es correcta. Un documento de prueba puede ser incorrecto por varias razones. Si es incorrecto, debe corregirse y la prueba se debe volver a ejecutar.

[9] "unscripted testing".
[10] contrato de la prueba es la traducción de "test charter"
[11] Antes "pruebas programadas".
[12] En el glosario del ISTQB no se ha definido el término "prueba fugada o prueba escapada", cuya traducción sería "escaped test". Sin embargo sí existe el término "defecto escapado o fugado" traducción del término "escaped defect".

Dado que los cambios en la base de prueba y en el objeto de prueba pueden hacer que un caso de prueba sea incorrecto, aunque se haya ejecutado anteriormente con éxito muchas veces, los probadores deben permanecer alerta ante la posibilidad de que los resultados observados se deban a una prueba incorrecta.

Durante la ejecución de las pruebas, los resultados de cada prueba se deben registrar de forma adecuada. Las pruebas que se hayan ejecutado pero cuyos resultados no se hayan registrado pueden tener que repetirse para identificar su resultado correcto, lo que puede dar lugar a ineficiencias y retrasos. (Cabe observar que un registro adecuado puede abordar los problemas de cobertura y repetibilidad asociados a técnicas de prueba como las pruebas exploratorias). Dado que el objeto de prueba, los productos de soporte de pruebas y los entornos de prueba pueden evolucionar, en el registro deben identificarse las versiones específicas probadas así como las configuraciones específicas del entorno. El registro en la bitácora de prueba establece un registro cronológico de información relevante sobre la ejecución de las pruebas.

El registro de los resultados es aplicable tanto a pruebas individuales como a actividades y eventos. Cada prueba debe ser identificada de forma unívoca y su estado registrado a medida que tiene lugar la ejecución de las pruebas. Cualquier evento que pueda afectar a la ejecución de las pruebas debe registrarse. Debe registrarse información suficiente para medir la cobertura de las pruebas y documentarse los motivos de los retrasos e interrupciones de las pruebas. Además, se debe registrar la información para dar soporte al control de la prueba, la gestión de la información sobre el avance de las pruebas, la medición de los criterios de salida y la mejora del proceso de pruebas.

El registro puede variar en función del nivel de prueba y la estrategia de prueba. Así por ejemplo, si se están llevando a cabo pruebas de componente automatizadas, las pruebas automatizadas deberían generar la mayor parte de la información de registro. En caso de que se realicen pruebas manuales, el Analista de Pruebas registrará la información sobre la ejecución de las pruebas, a menudo en una herramienta de gestión de pruebas que permitirá hacer el seguimiento de la información de ejecución de las pruebas. En algunos casos, como en la implementación de las pruebas, la cantidad de información registrada sobre la ejecución de las pruebas dependerá de requisitos normativos o requisitos de auditoría.

En algunos casos, los usuarios o clientes pueden participar en la ejecución de las pruebas. Esto puede servir para aumentar su confianza en el sistema, si bien con ello se está presuponiendo que las pruebas van a detectar pocos defectos. Este supuesto no suele ser válido en los niveles de prueba tempranos, pero sí puede serlo durante las pruebas de aceptación.

A continuación se citan algunas áreas específicas que deben tenerse en cuenta durante la ejecución de las pruebas:

- Informar sobre y explorar peculiaridades “irrelevantes”. Las observaciones o resultados que pueden parecer irrelevantes son a veces indicadores de defectos que (como los icebergs) se esconden bajo la superficie.
- Comprobar que el producto no está haciendo lo no que debería hacer. Comprobar que el producto hace lo que debería hacer es un foco habitual de las pruebas, pero el Analista de Pruebas también debe asegurarse de que el producto no tenga un mal comportamiento y que no haga cosas que no debería hacer (por ejemplo, funciones adicionales no deseadas).
- Construir el juego de pruebas y esperar que crezca y cambie con el tiempo. El código evolucionará y se deberán implementar pruebas adicionales para cubrir las nuevas funcionalidades así como para comprobar regresiones en otras áreas del software. Durante la ejecución, con frecuencia, se detectan vacíos en las pruebas. La construcción del juego de pruebas es un proceso continuo.
- Tomar notas de cara al siguiente esfuerzo de prueba. Las tareas de prueba no concluyen cuando el software se entrega al usuario o se distribuye comercialmente. Seguramente se creará una nueva versión o una nueva entrega del software por lo cual se deberían almacenar y transferir los conocimientos a los probadores encargados del siguiente esfuerzo de prueba.
- No asumir que se vaya a repetir todas las pruebas manuales. Es poco realista esperar que se repitan todas las pruebas manuales. Si se sospecha de la presencia de un problema, el Analista

de Pruebas debe investigarlo y tomar nota, en lugar de asumir que se detectará en una ejecución posterior de los casos de prueba.
- Extraer datos de la herramienta de seguimiento de defectos para casos de prueba adicionales. Considerar la creación de casos de prueba para los defectos detectados durante las pruebas sin guion o exploratorias y añadirlos al juego de pruebas de regresión.
- Detectar los defectos antes de las pruebas de regresión. A menudo el tiempo para las pruebas de regresión es limitado y la detección de fallos durante las pruebas de regresión puede dar lugar a retrasos en los plazos. Normalmente, las pruebas de regresión no localizan muchos defectos, fundamentalmente porque se trata de pruebas que han sido utilizadas anteriormente (por ejemplo, en una versión anterior del mismo software), y los defectos ya deberían haberse detectado en dichas utilizaciones anteriores. Esto no significa que las pruebas de regresión deban eliminarse sin más, aunque sí que la eficiencia de las pruebas de regresión, en términos de su capacidad para detectar nuevos defectos, es menor que la de otras pruebas.

1.8 Evaluación de los Criterios de Salida y Gestión de Información de la Prueba

Desde el punto de vista del proceso de prueba, la monitorización del avance de las pruebas supone garantizar que se recopila la información adecuada para dar soporte a los requisitos de gestión de información de la prueba. Esto incluye medir el avance logrado hasta su compleción. Cuando se definen los criterios de salida en las etapas de planificación, puede establecerse una división entre criterios opcionales y criterios obligatorios. Por ejemplo, los criterios pueden indicar que “no debe haber ningún bug abierto de Prioridad 1 o Prioridad 2” y que debería alcanzarse “una tasa[13] de paso del 95% para todos los casos de prueba”. En este caso, el incumplimiento de los criterios “obligatorios” debería provocar el fallo de los criterios de salida, mientras que una frecuencia de paso del 93% permitiría que el proyecto pasara al siguiente nivel. Los criterios de salida deben estar claramente definidos para poder ser evaluados de una forma objetiva.

El Analista de Pruebas es responsable de suministrar la información que utilizará el Jefe de Pruebas para evaluar el avance en la consecución de los criterios de salida y para garantizar la exactitud de los datos. Si, por ejemplo, el sistema de gestión de pruebas establece los siguientes códigos de estado para la compleción de un caso de prueba:
- Pasado.
- Fallado.
- Pasado con excepción.

entonces el Analista de Pruebas debe ser muy claro sobre el significado de cada uno de estos estados y debe aplicarlos de una manera consistente. ¿El estado “pasado con excepción” significa que se ha encontrado un defecto pero que éste no afecta a la funcionalidad del sistema? ¿Qué ocurre con un defecto de usabilidad que confunde al usuario? Si la frecuencia de paso constituye un criterio de salida “obligatorio”, calificar un caso de prueba como “fallado” en lugar de como “pasado con excepción” se convierte en un factor crítico. También deben tenerse en cuenta los casos de prueba marcados como “fallados” cuya causa de fallo no es un defecto (por ejemplo, el entorno de pruebas no se ha configurado correctamente). Si existe confusión en lo que respecta a las métricas monitorizadas o al uso de los valores de estado, el Analista de Pruebas debe aclarar estos aspectos con el Jefe de Pruebas para que la información pueda objeto de seguimiento de una forma exacta y consistente a lo largo de todo el proyecto.

No es raro que el Analista de Pruebas tenga que elaborar informes de estado durante los ciclos de prueba o participar en la elaboración del informe final al término de las pruebas. Para ello puede ser necesario recopilar métricas de los sistemas de gestión de defectos y pruebas, además de evaluar la cobertura y el avance generales. El Analista de Pruebas debe ser capaz de utilizar las herramientas de gestión de información y de facilitar la información requerida para que el Jefe de Pruebas pueda extraer la información necesaria.

[13] Porcentaje de casos de prueba que han pasado la prueba.

1.9 Actividades de Cierre de Pruebas

Una vez que se ha determinado que la ejecución de pruebas está completa, se deben registrar las salidas clave del esfuerzo de prueba y, bien, comunicarlas a la persona correspondiente o archivarlas. En general, estas actividades se denominan actividades de cierre de pruebas. Cabe esperar que el Analista de Pruebas esté involucrado en la entrega de productos de trabajo a aquellos que puedan necesitarlos. Por ejemplo, los defectos conocidos que han sido diferidos o aceptados deben ser comunicados a aquellas personas que usen y soporten el uso del sistema. Las pruebas y los entornos de prueba deben facilitarse a los responsables de las pruebas de mantenimiento. Otro producto de trabajo puede ser un conjunto de pruebas de regresión (tanto automatizadas como manuales). La información sobre los productos de trabajo de las pruebas debe estar claramente documentada, incluyendo enlaces adecuados, debiendo concederse los correspondientes derechos de acceso.

También cabe esperar que el Analista de Pruebas participe en reuniones retrospectivas ("lecciones aprendidas") donde pueden documentarse las lecciones importantes (tanto del propio proyecto de prueba como de todo el ciclo de vida de desarrollo de software) y pueden establecerse planes para reforzar lo "bueno" y eliminar, o por lo menos controlar, lo "malo". El Analista de Pruebas constituye una fuente de información reconocida para este tipo de reuniones y debe participar en ellas si debe recopilarse información válida para la mejora del proceso. En caso de que solo el Jefe de Pruebas asista a estas reuniones, el Analista de Pruebas deberá trasladar la información pertinente al Jefe de Pruebas de forma que presente una imagen fiel del proyecto.

Se deben archivar los resultados, registros, informes y demás documentos y productos de trabajo en el sistema de gestión de la configuración. Esta tarea a menudo compete al Analista de Pruebas y constituye una importante actividad de cierre, especialmente si un proyecto futuro requiere el uso de esta información.

Si bien el Jefe de Pruebas es normalmente el que determina qué información se debe archivar, el Analista de Pruebas también debe plantearse qué información sería necesaria si el proyecto tuviera que volver a realizarse en un futuro. Establecer esta información al término de un proyecto puede ahorrar meses de esfuerzo si el proyecto se vuelve a iniciar en un futuro o con otro equipo.

2. Gestión de Pruebas: Responsabilidades del Analista de Pruebas - 90 minutos

Palabras Clave

riesgo de producto, análisis de riesgos, identificación de riesgos, nivel de riesgo, gestión de riesgos, mitigación de riesgos, pruebas basadas en riesgos, monitorización de pruebas, estrategia de pruebas

Objetivos de aprendizaje de la Gestión de pruebas: Responsabilidades del Analista de Pruebas

2.2 Monitorización y Control del Avance de las Pruebas

AP-2.2.1 (K2) Explicar los tipos de información que deben ser objeto de seguimiento durante las pruebas para permitir una correcta monitorización y control del proyecto.

2.3 Pruebas Distribuidas, Externalizadas e Internalizadas

AP-2.3.1 (K2) Poner ejemplos de buenas prácticas de comunicación durante el trabajo en un entorno de pruebas de 24 horas.

2.4 Tareas del Analista de Pruebas en las Pruebas Basadas en Riesgos

AP-2.4.1 (K3) Para un escenario de proyecto dado, participar en la identificación de riesgos, llevar a cabo una evaluación de riesgos y proponer medidas adecuadas para la mitigación de riesgos.

2.1 Introducción

Si bien hay muchas áreas en las que el Analista de Pruebas interactúa con el Jefe de Pruebas y suministra datos, esta sección se centra en las áreas específicas del proceso de pruebas para las que el Analista de Pruebas se constituye colaborador principal. Se espera que el Jefe de Pruebas solicite la información necesaria al Analista de Pruebas.

2.2 Monitorización y Control del Avance de la Prueba

Hay cinco dimensiones primarias a partir de las cuales se monitoriza el avance de la prueba:

- Riesgos de producto (riesgos de calidad).
- Defectos.
- Pruebas.
- Cobertura.
- Confianza.

El Analista de Pruebas suele ser el encargado de medir y reportar los riesgos de producto, los defectos, las pruebas y la cobertura de una forma específica durante el proyecto o en operación. La confianza, aunque puede medirse a través de estudios, normalmente se reporta de forma subjetiva. La recopilación de la información necesaria para dar soporte a estas métricas forma parte del trabajo diario del Analista de Pruebas. Es importante recordar que la exactitud de estos datos es fundamental, dado que los datos inexactos generarán información inexacta, lo que puede dar lugar a conclusiones erróneas. En el peor de los casos, los datos inexactos darán lugar a decisiones de gestión incorrectas y dañarán la credibilidad del equipo de pruebas.

Cuándo se utiliza un enfoque de pruebas basado en riesgos, el Analista de Pruebas debe hacer un seguimiento de:

- Qué riesgos han mitigado las pruebas.
- Qué riesgos se considera que no se han mitigado.

Con frecuencia, el seguimiento de la mitigación de riesgos se lleva a cabo mediante una herramienta que también monitoriza la compleción de las pruebas (por ejemplo, herramientas de gestión de pruebas). Esto requiere que los riesgos identificados estén asociados[14] a las condiciones de prueba que, a su vez, están asociadas a los casos de prueba que, una vez ejecutados y habiendo pasado, mitigarán los riesgos. De esta forma, la información sobre la mitigación de riesgos se actualiza automáticamente a medida que se actualizan los casos de prueba. Esto se puede hacer para pruebas tanto manuales como automatizadas.

El seguimiento de los defectos normalmente se realiza mediante una herramienta de seguimiento de defectos. A medida que los defectos se registran, también se registra información de clasificación específica de cada defecto. Esta información se utiliza para generar tendencias y gráficos que indican el avance de la prueba y la calidad del software. La información de clasificación se aborda con más detalle en el capítulo sobre Gestión de Defectos. El ciclo de vida puede afectar a la cantidad de documentación de defectos registrada y a los métodos empleados para el registro de la información.

A medida que las pruebas se llevan a cabo, debe registrase información sobre el estado del caso de prueba. Con frecuencia, esto suele hacerse mediante una herramienta de gestión de pruebas, aunque también puede hacerse manualmente si fuera necesario. La información de los casos de prueba puede incluir:

- Estado de creación del caso de prueba (por ejemplo, diseñado, revisado, etc.).
- Estado de ejecución del caso de prueba (por ejemplo, pasado, fallado, bloqueado, obviado, etc.).
- Información sobre la ejecución del caso de prueba (por ejemplo, fecha y hora, nombre del probador, datos utilizados, etc.).
- Artefactos de la ejecución del caso de prueba (por ejemplo, capturas de pantalla, registros asociados, etc.).

[14] En este caso la asociación es una correspondencia (relación o una traza) entre el par de objetos.

De la misma forma que ocurre con los elementos de riesgo identificados, los casos de prueba deben estar asociados[15] a los elementos de requisitos que van a probar. Es importante que el Analista de Pruebas recuerde que si hay un caso de prueba A asociado a un requisito A, y este es el único caso de prueba asociado a dicho requisito, entonces, si el caso de prueba A se ejecuta y pasa, se considerará que el requisito A se ha cumplido. Esto puede o no ser correcto. En muchos casos, se requieren más casos de prueba para probar minuciosamente un requisito, pero dadas las limitaciones de tiempo, en la práctica solo se crea un subconjunto de dichas pruebas. Así por ejemplo, si se requieren 20 casos de prueba para probar minuciosamente la implementación de un requisito, pero sólo se crean y se ejecutan 10, entonces la información sobre la cobertura del requisito indicará que hay un 100% de cobertura cuando en realidad solo se ha alcanzado un 50% de cobertura. El seguimiento preciso de la cobertura, sumado a un seguimiento del estado revisado de los propios requisitos, puede utilizarse como una medición de la confianza.

La cantidad (y el nivel de detalle) de la información a registrar dependerá de varios factores, incluyendo el modelo de ciclo de vida de desarrollo de software. Así por ejemplo, en proyectos Ágiles normalmente se registrará menos información sobre el estado dada la estrecha interacción del equipo y la mayor comunicación cara a cara.

2.3 Pruebas Distribuidas, Externalizadas e Internalizadas

En muchos casos, no todo el esfuerzo de prueba lo realiza un único equipo de pruebas, compuesto por compañeros de trabajo del resto del equipo de proyecto, en una única y misma ubicación que el resto del equipo del proyecto. Si el esfuerzo de prueba tiene lugar en múltiples ubicaciones, ese esfuerzo de prueba se puede hablar de un esfuerzo distribuido. Si tiene lugar en una única ubicación, puede hablarse de esfuerzo centralizado. Si el esfuerzo de prueba se realiza en una o más ubicaciones por personas que no son compañeros empleados del resto del equipo de proyecto y que no se encuentran en la misma ubicación que el equipo del proyecto, ese esfuerzo de prueba puede denominarse esfuerzo externalizado. Si el esfuerzo de prueba lo realizan personas que están ubicadas en el mismo lugar que el equipo del proyecto pero que no son compañeros de trabajo, el esfuerzo de prueba puede denominarse esfuerzo internalizado.

Si el Analista de Pruebas trabaja en un proyecto en el que parte del equipo de pruebas está distribuido entre varias ubicaciones, o incluso entre varias empresas, deberá prestar una atención especial a la comunicación y transferencia de información efectivos. En algunas empresas se sigue un modelo de “24 horas de prueba”, según el cual el equipo en una zona horaria debe entregar el trabajo al equipo ubicado en otra zona horaria para que las pruebas puedan continuar a lo largo de las veinticuatro horas del día. Esto requiere una planificación especial por parte del Analista de Pruebas encargado de entregar y recibir el trabajo. Una buena planificación es importante para conocer las responsabilidades, pero es fundamental para garantizar que se dispone de la información adecuada.

Cuando no hay posibilidad de comunicación oral, la comunicación por escrito debe ser suficiente. Esto significa que se debe recurrir al correo electrónico, a los informes de estado y al uso efectivo de las herramientas de gestión de pruebas y seguimiento de defectos. Si una herramienta de gestión de pruebas permite asignar pruebas a individuos, también puede emplearse como herramienta de planificación y constituir una forma sencilla de transferir trabajo entre personas. Los defectos que se registran correctamente pueden transferirse a otros colaboradores para su seguimiento en caso necesario. El uso efectivo de estos sistemas de comunicación es vital para una organización que no puede depender de una interacción personal diaria.

2.4 Tareas del Analista de Pruebas en las Pruebas Basadas en Riesgos

2.4.1 Resumen

El Jefe de Pruebas, a menudo, tiene la responsabilidad general de establecer y gestionar una estrategia de prueba basada en riesgos. Normalmente, el Jefe de Pruebas pedirá Analista de Pruebas que se

[15] En este caso la asociación es una correspondencia (relación o una traza) entre el par de objetos.

involucre para garantizar que el enfoque de pruebas basado en riesgos se implemente de forma correcta.

El Analista de Pruebas debería estar involucrado en las siguientes tareas de pruebas basadas en riesgos:

- Identificación del riesgo.
- Evaluación del riesgo.
- Mitigación del riesgo.

Estas tareas se llevan a cabo de manera iterativa durante todo el ciclo de vida del proyecto para afrontar los riesgos emergentes, modificar prioridades así como evaluar y comunicar periódicamente el estado de los riesgos.

Los Analistas de Pruebas deben trabajar dentro del marco de trabajo de las pruebas basadas en riesgos establecido por el Jefe de Pruebas para el proyecto. Éstos deberían contribuir con su conocimiento de los riesgos del dominio del negocio que son inherentes al proyecto, tales como los riesgos asociados a la seguridad, asuntos relativos al negocio y de carácter económico así como factores políticos.

2.4.2 Identificación del Riesgo

Si se recurre a la muestra de implicados más amplia posible, el proceso de identificación de riesgos tendrá más posibilidades de detectar el mayor número posible de riesgos significativos. Dado que, a menudo, los Analistas de Pruebas poseen un conocimiento único en lo que respecta al ámbito de negocio específico del sistema objeto de prueba, resultan particularmente idóneos para llevar a cabo entrevistas expertas a usuarios y expertos del dominio, realizar evaluaciones independientes, utilizar y facilitar el uso de plantillas de riesgos, realizar talleres de riesgos, organizar sesiones de tormenta de ideas con usuarios potenciales y reales, definir listas de comprobación de pruebas y recurrir a experiencias pasadas en proyectos o sistemas similares. En particular, el Analista de Pruebas debe trabajar en estrecha colaboración con los usuarios y demás expertos del dominio para establecer las áreas de riesgos de negocio que deberían abordarse durante las pruebas. El Analista de Pruebas también puede resultar especialmente útil a la hora de identificar los posibles efectos del riesgo en los usuarios e implicados.

Una muestra de riesgos que pueden ser identificados en un proyecto incluye:

- Problemas de exactitud con la funcionalidad del software, por ejemplo, cálculos incorrectos.
- Problemas de usabilidad, por ejemplo, combinaciones de teclas insuficientes.
- Problemas relativos a la capacidad de ser aprendido[16], por ejemplo, ausencia de instrucciones para el usuario en puntos de decisión clave.

En el Capítulo 4 de este programa de estudio se abordan las pruebas de características de calidad específicas.

2.4.3 Evaluación del Riesgo

Mientras que la identificación de riesgos se refiere a la identificación del máximo número de riesgos pertinentes, la evaluación del riesgo es el estudio de los riesgos identificados. En particular, consiste en clasificar los riesgos y establecer la probabilidad y el impacto asociados a cada uno de ellos.

Determinar el nivel de riesgo normalmente implica la evaluación, para cada elemento de riesgo, de la probabilidad de ocurrencia y del impacto en caso de ocurrencia. La probabilidad de ocurrencia se representa normalmente como la probabilidad de que el problema potencial pueda existir en el sistema objeto de prueba y que se observará cuando el sistema se encuentre en producción. En otras palabras, proviene del riesgo técnico. El Analista de Pruebas Técnicas debe contribuir a detectar y conocer el nivel de riesgo técnico potencial para cada elemento de riesgo, mientras que el Analista de Pruebas contribuye a conocer el impacto de negocio que podría tener el problema en caso de producirse.

[16] Antes aprendibilidad.

El impacto en caso de ocurrencia se interpreta, a menudo, como la severidad del efecto para los usuarios, clientes y demás implicados. En otras palabras, proviene del riesgo de negocio. El Analista de Pruebas debería contribuir a la identificación y evaluación del impacto potencial en el dominio de negocio o en el usuario para cada elemento de riesgo. Entre los factores que influyen en el riesgo de negocio se encuentran:

- Frecuencia del uso de la prestación afectada.
- Pérdida de negocio.
- Posibles pérdidas o responsabilidad financieras, medioambientales o sociales.
- Sanciones legales civiles o penales.
- Problemas de seguridad.
- Multas, pérdida de licencia.
- Carencia de soluciones temporales.
- Visibilidad de la prestación.
- Visibilidad de fallos que dan lugar a publicidad negativa y posibles daños en la reputación.
- Pérdida de clientes.

Dada la información disponible sobre los riesgos, el Analista de Pruebas tiene que establecer los niveles de riesgo de negocio de conformidad con las directrices establecidas por el Jefe de Pruebas. Dichos niveles pueden clasificarse mediante términos (por ejemplo, bajo, medio o alto) o cifras. A menos que exista una forma de medir de manera objetiva el riesgo en una escala definida, no puede tratarse de una medida cuantitativa verdadera. Normalmente es muy difícil medir exactamente la probabilidad y el coste/impacto, por lo que el nivel de riesgo normalmente se determina de forma cualitativa.

Se pueden asignar valores numéricos a un valor cualitativo, pero esto no significa que se trate de una verdadera medida cuantitativa. Así por ejemplo, el Jefe de Pruebas puede establecer que el riesgo de negocio debe categorizarse utilizando valores del 1 al 10, siendo 1 el impacto para el negocio más alto, y por lo tanto de mayor riesgo. Una vez asignada la probabilidad (la evaluación del riesgo técnico) y el impacto (la evaluación del riesgo de negocio), estos valores podrán multiplicarse para establecer el riesgo general asociado a la clasificación de cada elemento de riesgo. La calificación general se utiliza a continuación para priorizar las actividades de mitigación de riesgos. Algunos modelos de pruebas basados en riesgos, como PRISMA® [vanVeenendaal12], no combinan los valores de riesgo, lo que hace que el enfoque de pruebas pueda abordar los riesgos técnicos y de negocio por separado[17].

2.4.4 Mitigación del Riesgo

Durante el proyecto, los Analistas de Pruebas deberían intentar:

- Reducir el riesgo de producto empleando casos de prueba bien diseñados que demuestren de una forma inequívoca[18] si los elementos de prueba pasan o fallan, y participando en revisiones de artefactos software como requisitos, diseños y documentación del usuario.
- Implementar actividades adecuadas de mitigación de riesgos identificadas en la estrategia de prueba y en el plan de prueba.
- Volver a evaluar los riesgos conocidos en función de la información adicional recopilada a medida que el proyecto avanza, ajustando la probabilidad, el impacto, o ambos, según corresponda.
- Reconocer nuevos riesgos identificados por la información obtenida durante las pruebas.

Cuando se habla del riesgo de producto (de calidad), las pruebas son una forma de mitigar dichos riesgos. Al detectar defectos, los probadores reducen el riesgo aportando el conocimiento de la presencia de defectos y las oportunidades para tratarlos antes de la entrega. Si los probadores no detectan defectos, las pruebas reducen el riesgo garantizando que, en determinadas condiciones (por ejemplo, las condiciones probadas), el sistema funciona correctamente. Los Analistas de Pruebas ayudan a determinar las opciones para la mitigación de riesgos explorando oportunidades para recopilar datos de prueba exactos, creando y probando escenarios de usuario realistas y realizando o supervisando estudios de usabilidad.

[17] De forma independiente.
[18] Sin ambigüedad.

2.4.4.1 Priorización de las Pruebas

El nivel de riesgo también se utiliza para priorizar las pruebas. Un Analista de Pruebas puede establecer que existe un alto riesgo en el área de la exactitud en el ámbito de las transacciones en un sistema de contabilidad. En consecuencia, para mitigar este riesgo, el probador puede colaborar con expertos del dominio del negocio con el fin de reunir un conjunto sólido de datos de muestra que se puedan procesar y verificar en lo que respecta a la exactitud. De manera similar, un Analista de Pruebas también puede determinar que los problemas de usabilidad constituyen un riesgo importante para un nuevo producto. En lugar de esperar a que una prueba de aceptación de usuario identifique cualquier problema, el Analista de Pruebas podría priorizar una prueba temprana de usabilidad durante el nivel de integración para ayudar a identificar y solucionar problemas de usabilidad de forma temprana en las pruebas. Esta priorización debe considerarse lo antes posible en las fases de planificación de manera que el calendario pueda ajustarse a las pruebas necesarias en el momento oportuno.

En algunos casos, todas las pruebas con mayor riesgo se ejecutan antes que las pruebas de menor riesgo, y las pruebas se ejecutan en estricto orden de riesgo (a menudo denominado “profundidad primero[19]”), en otros casos, se utiliza un enfoque de muestreo para seleccionar una muestra de pruebas de entre todos los riesgos identificados, utilizando el riesgo para ponderar la selección al tiempo que se garantiza la cobertura de cada riesgo, al menos, una vez (a menudo denominado “amplitud primero[20]”).

Tanto si las pruebas basadas en riesgos se realizan primero en profundidad o primero en amplitud, es posible que el tiempo asignado para las pruebas se consuma sin que se hayan ejecutado todas las pruebas. Las pruebas basadas en riesgos permiten a los probadores informar a la dirección sobre el nivel de riesgo restante en un punto dado, y permite a la dirección decidir sobre si ampliar las pruebas o transferir el riesgo restante a los usuarios, clientes, atención al cliente/soporte técnico y/o personal operativo.

2.4.4.2 Ajuste de las Pruebas para Futuros Ciclos de Prueba

La evaluación del riesgo no es una actividad que se realice una única vez antes de la puesta en marcha de la implementación de la prueba, se trata de un proceso continuo. Todos los ciclos de prueba planificados para el futuro deberían someterse a un nuevo análisis de riesgos que deberá tener en cuenta factores como:

- Cualquier riesgo de producto nuevo o modificado de forma significativa.
- Áreas inestables o proclives a los defectos detectadas durante las pruebas.
- Riesgos derivados de defectos corregidos.
- Defectos típicos descubiertos durante las pruebas.
- Áreas no probadas lo suficiente (baja cobertura de prueba).

Si se asigna más tiempo a las pruebas se podría ampliar la cobertura de riesgos a áreas de menor riesgo.

[19] profundidad primero es la traducción del término “depth-first”. Otra traducción podría ser “primero en profundidad”.
[20] amplitud primero es la traducción del término “depth-first”. Otra traducción podría ser “primero en amplitud”.

3. Técnicas de Prueba - 825 minutos

Palabras Clave

análisis de valores frontera (AVF), representación causa-efecto, pruebas basadas en listas de comprobación, método del árbol de clasificación, pruebas combinatorias, pruebas de tabla de decisión, taxonomía de defectos, técnica basada en defectos, análisis de dominio, predicción de errores, segmentación de equivalencia, técnica basada en la experiencia, pruebas exploratorias, arreglo ortogonal, pruebas de arreglos ortogonales, pruebas de a pares de elementos, pruebas basadas en requisitos, técnica basada en la especificación, pruebas de transición de estado, contrato de la prueba, pruebas de caso de uso, pruebas de historias de usuario

Objetivos de Aprendizaje de las Técnicas de prueba

3.2 Técnicas Basadas en la Especificación

AP-3.2.1 (K2) Explicar el uso de gráficos causa-efecto.

AP-3.2.2 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de segmentación de equivalencia para alcanzar un nivel de cobertura establecido.

AP-3.2.2 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de análisis de valores frontera para alcanzar un nivel de cobertura establecido.

AP-3.2.4 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de tabla de decisión para alcanzar un nivel de cobertura establecido.

AP-3.2.5 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de transición de estado para alcanzar un nivel de cobertura establecido.

AP-3.2.6 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de a pares de elementos para alcanzar un nivel de cobertura establecido.

AP-3.2.7 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de árbol de clasificación para alcanzar un nivel de cobertura establecido.

AP-3.2.8 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de caso de uso para alcanzar un nivel de cobertura establecido.

AP-3.2.9 (K2) Explicar cómo se utilizan las historias de usuario para orientar las pruebas en un proyecto Ágil.

AP-3.2.10 (K3) Redactar casos de prueba a partir de un elemento de especificación dado aplicando la técnica de diseño de pruebas de análisis de dominio para alcanzar un nivel de cobertura establecido.

AP-3.2.11 (K4) Analizar un sistema, o su especificación de requisitos, para determinar qué tipos de defectos es probable que se encuentren y seleccionar la técnica basada en la especificación más apropiada.

3.3 Técnicas Basadas en Defectos

AP-3.3.1 (K2) Describir la aplicación de técnicas de prueba basadas en defectos y diferenciar su uso de las técnicas basadas en la especificación.

AP-3.3.2 (K4) analizar una taxonomía de defectos dada para estudiar su aplicabilidad en una situación dada empleando criterios para una correcta taxonomía.

3.4 Técnicas Basadas en la Experiencia

AP-3.4.1 (K2) Explicar los principios de las técnicas basadas en la experiencia, y las ventajas e inconvenientes frente a las técnicas basadas en la especificación y las técnicas basadas en defectos.

AP-3.4.2 (K3) Dado un escenario, especificar las pruebas exploratorias y explicar cómo se pueden informar los resultados.

AP-3.4.3 (K4) Dada una situación de proyecto, establecer qué técnica basada en la especificación, basada en defectos o basada en la experiencia debería aplicarse para alcanzar objetivos específicos.

3.1 Introducción

Las técnicas de diseño de prueba consideradas en este capítulo se dividen en las siguientes categorías:

- Basadas en la especificación (o basadas en el comportamiento o de caja negra).
- Basadas en defectos.
- Basadas en la experiencia.

Estas técnicas son complementarias y pueden utilizarse como sea apropiado para cualquier actividad de prueba, con independencia del nivel de prueba que se esté realizando.

Observe que las tres categorías de técnicas pueden emplearse para probar características de calidad, tanto funcionales como no funcionales. En el siguiente capítulo se tratarán las pruebas de características no funcionales.

Las técnicas de diseño de pruebas abordadas en estas secciones se centran principalmente en la determinación de datos de prueba óptimos (por ejemplo, partición de equivalencia) o la obtención de secuencias de pruebas (por ejemplo, modelos de estado). Es frecuente combinar varias técnicas para crear casos de prueba completos.

3.2 Técnicas Basadas en la Especificación

Las técnicas basadas en la especificación se aplican a las condiciones de prueba para obtener casos de prueba basados en el análisis de las bases de prueba de un componente o sistema sin hacer referencia a su estructura interna.

Algunas características comunes de las técnicas basadas en la especificación son:

- Durante el diseño de la prueba se crean modelos, por ejemplo, diagramas de transición de estado y tablas de decisión, de acuerdo con a la técnica de prueba.
- Las condiciones de prueba se obtienen, de forma sistemática, de estos modelos.

Algunas técnicas también aportan criterios de cobertura que pueden utilizarse para medir las tareas de diseño y ejecución de las pruebas. El cumplimiento íntegro de los criterios de cobertura no significa que el conjunto de pruebas esté completo, sino más bien que el modelo ya no sugiere ninguna prueba adicional para aumentar la cobertura basándose en esa técnica.

Normalmente, las pruebas basadas en la especificación se basan en los documentos de requisitos del sistema. Dado que en la especificación de los requisitos debería especificar cómo se debe comportar el sistema, en especial en el área de la funcionalidad, crear pruebas a partir de los requisitos, a menudo, forma parte de las pruebas de comportamiento del sistema. En algunos casos puede no haber requisitos documentados, sino requisitos implícitos tales como sustituir la funcionalidad de un sistema legado.

Existen varias técnicas de prueba basadas en la especificación. Estas técnicas están dirigidas a distintos tipos de software y escenarios. Las secciones a continuación muestran la aplicabilidad de cada técnica, algunas limitaciones y dificultades con las que puede encontrarse el Analista de Pruebas, el método para medir la cobertura de la prueba y el tipo de defectos que se busca.

3.2.1 Segmentación de Equivalencia

La segmentación de equivalencia (SE) sirve para reducir el número de casos de prueba necesario para probar de manera efectiva el tratamiento de entradas, salidas, valores internos y valores temporales. La segmentación permite crear clases de equivalencia (a menudo denominadas particiones de equivalencia) que constan de conjuntos de valores que se procesan de la misma forma. Al seleccionar

un valor representante de una partición, se asume que todos los elementos incluidos en dicha partición están cubiertos.

Aplicabilidad
Esta técnica es aplicable a cualquier nivel de prueba y es adecuada cuando se espera que todos los miembros de un conjunto de valores se van a tratar de la misma forma y cuando los conjuntos de valores utilizados por la aplicación no interactúan. La selección de los conjuntos de valores es aplicable a las particiones válidas y no válidas (es decir, aquellas particiones que incluyan valores que deberían considerarse no válidos para el software objeto de prueba). Esta técnica más fuerte si se utiliza en combinación con un análisis de los valores frontera que permita ampliar los valores de prueba para incluir aquellos ubicados en los bordes[21] de las particiones. Se trata de una técnica muy utilizada en las pruebas de humo de nuevas construcciones[22] o entregas ya que permite determinar rápidamente si la funcionalidad básica funciona.

Limitaciones/Dificultades
Si la suposición es incorrecta y los valores incluidos en la partición no se tratan exactamente de la misma forma, esta técnica puede obviar algunos defectos. También es importante seleccionar las particiones con atención. Así por ejemplo, un campo de entrada que acepta números positivos y negativos podría probarse mejor como dos particiones válidas, una para los números positivos y otra para los negativos, dada la probabilidad de que estos se traten de forma distinta. Dependiendo de si se permite o no el valor cero, éste también podría convertirse en una nueva partición. Es importante que el Analista de Pruebas conozca cuál es el procesamiento subyacente para establecer la mejor partición de los valores.

Cobertura
La cobertura se determina dividiendo el número de particiones para las que se ha probado un valor entre el número de particiones identificadas. El hecho de utilizar varios valores para una única partición no aumenta el porcentaje de cobertura.

Tipos de defectos
Esta técnica detecta defectos funcionales en el tratamiento de distintos valores de datos.

3.2.2 Análisis de Valores Frontera

El análisis de valores frontera (AVF) se utiliza para probar los valores que existen en las fronteras de las particiones de equivalencia ordenadas. Hay dos formas de abordar el AVF: pruebas de dos o de tres valores. En las pruebas de dos valores se utilizan el valor frontera (en la frontera) y el valor justo posterior al límite (sumando el mínimo incremento posible). Por ejemplo, si la partición incluye los valores de 1 a 10 en incrementos de 0,5, los valores de la prueba de dos valores para la frontera superior serían 10 y 10,5. Los valores de prueba para la frontera inferior serían 1 y 0,5. Las fronteras están definidas por los valores máximo y mínimo de la partición de equivalencia definida.

Para las pruebas de valores frontera con tres valores, se utilizan el valor anterior a la frontera, el valor sobre la frontera y el valor posterior a la frontera. En el ejemplo anterior, los valores de las pruebas para la frontera superior serían 9,5, 10 y 10,5. Los valores de prueba para la frontera inferior serían 1,5, 1 y 0,5. La decisión sobre si utilizar dos o tres valores límite debería basarse en el riesgo asociado al elemento que se está probando, siendo el enfoque de tres valores límite el aplicado en el caso de elementos con más riesgo.

Aplicabilidad
Esta técnica es aplicable en cualquier nivel de prueba y es adecuada cuando las particiones de equivalencia están ordenadas. El orden es necesario debido al concepto de estar en o por encima de la frontera. Por ejemplo, un rango de números constituye una partición ordenada. Una partición que consta de objetos todos rectangulares no es una partición ordenada y no tiene valores frontera. Además de los rangos de números, el análisis de valores frontera puede aplicarse a lo siguiente:

- Atributos numéricos de variables no numéricas (por ejemplo, la longitud).

[21] Extremo de la partición.
[22] Construcción es la traducción del término “build”. En algunos casos se traduce como versión de un producto.

- Bucles, incluidos aquellos en casos de uso.
- Estructuras de datos almacenadas.
- Objetos físicos (incluyendo memoria).
- Actividades determinadas por el tiempo.

Limitaciones/Dificultades
Dado que la exactitud de esta técnica depende de la correcta identificación de las particiones de equivalencia, está sujeta a las mismas limitaciones y dificultades. El Analista de Pruebas también debe tener en cuenta los incrementos en los valores válidos y no válidos para poder establecer con exactitud los valores a probar. El análisis de valores frontera sólo se puede realizar con particiones ordenadas, pero esto no se limita a un rango de entradas válidas. Por ejemplo, al probar el número de celdas soportado por una hoja de cálculo, hay una partición que contiene el número de celdas hasta e incluyendo el máximo permitido de celdas (la frontera) y otra partición que empieza en una celda por encima del máximo (por encima de la frontera).

Cobertura
La cobertura se calcula dividiendo el número de condiciones frontera probadas entre el número de condiciones frontera identificadas (bien aplicando el método de dos valores o de tres valores). Esto nos dará el porcentaje de cobertura de las pruebas de valores frontera.

Tipos de defectos
El análisis de valores frontera detecta de manera fiable el desplazamiento o la omisión de fronteras, y puede detectar casos de fronteras adicionales. Esta técnica encuentra defectos relativos al tratamiento de los valores frontera, en particular errores con una lógica de **“menor que”** y **“mayor que”** (es decir, desplazamiento). También puede utilizarse para detectar defectos no funcionales, como por ejemplo la tolerancia de los límites de carga (por ejemplo, el sistema soporta 10.000 usuarios concurrentes).

3.2.3 Tablas de Decisión

Las tablas de decisión sirven para probar la interacción entre combinaciones de condiciones. Las tablas de decisión proporcionan un método claro para verificar las pruebas de todas las combinaciones de condiciones pertinentes y para verificar que el software sujeto a prueba realiza el tratamiento de todas las posibles combinaciones. El objetivo de las pruebas de tablas de decisión es asegurar que se prueban todas las combinaciones de condiciones, relaciones y restricciones. Cuando se intentan probar todas las combinaciones posibles, las tablas de decisión pueden resultar muy extensas. Una forma de reducir, de manera inteligente, el número de combinaciones de todas las posibles a aquellas que son “interesantes” es el método denominado pruebas de tablas de decisión colapsadas. Cuando se utiliza esta técnica, las combinaciones se reducen a aquellas que produzcan salidas distintas. Se eliminan las pruebas redundantes o las pruebas en las que no es posible la combinación de condiciones. La decisión de si utilizar tablas de decisión completas o tablas de decisión colapsadas normalmente depende del riesgo. [Copeland03]

Aplicabilidad
Esta técnica suele aplicarse en los niveles de pruebas de integración, pruebas de sistema y pruebas de aceptación. Dependiendo del código, también se puede utilizar en las pruebas de componente cuando un componente es responsable de un conjunto de lógica de decisión. Esta técnica es especialmente útil cuando los requisitos se presentan en forma de diagramas de flujo o tablas de reglas de negocio. Las tablas de decisión también son una técnica para la definición de requisitos, pudiendo algunas especificaciones de requisitos llegar ya en este formato. Incluso cuando los requisitos no se presentan en una forma tabular o de diagrama de flujo, normalmente las combinaciones de condiciones se encuentran en la narrativa. Cuando se diseñan tablas de decisión, es importante tener en cuenta las combinaciones de condiciones definidas y aquellas que, aunque no están expresamente definidas, existirán. Para diseñar una tabla de decisión válida, el probador debe ser capaz de establecer todos los resultados esperados para todas las combinaciones de condiciones a partir del oráculo de especificaciones o pruebas. La tabla de decisión sólo será una buena herramienta de diseño de pruebas si tiene en cuenta todas las condiciones que interactúan.

Limitaciones/Dificultades

Encontrar todas las condiciones que interactúan puede suponer un reto, especialmente cuando los requisitos no están bien definidos o no existen. No es raro elaborar un conjunto de condiciones y determinar que no se conoce el resultado esperado.

Cobertura
La cobertura de pruebas mínima para una tabla de decisión es disponer de un caso de prueba por cada columna. Este supuesto implica que no hay condiciones compuestas y que en una columna se han registrado todas las combinaciones de condiciones posibles. A la hora de determinar las pruebas a partir de una tabla de decisión, también es importante tener en cuenta que se deben probar todas las condiciones frontera. Estas condiciones frontera pueden dar lugar a un aumento del número de casos de prueba necesarios para probar el software de forma adecuada. El análisis de valores frontera y la segmentación de equivalencia son técnicas complementarias a la de tabla de decisión.

Tipos de defectos
Entre los defectos típicos se encuentra el procesamiento incorrecto basado en combinaciones específicas de condiciones que dan lugar a resultados no esperados. Durante la creación de las tablas de decisión, pueden encontrarse defectos en el documento de especificaciones. Los tipos de defectos más comunes son omisiones (no hay información sobre qué debería suceder en realidad en una situación dada) y contradicciones. Las pruebas también pueden identificar problemas con combinaciones de condiciones que no se tratan o que no se tratan bien.

3.2.4 Gráficos Causa-Efecto

Los gráficos causa-efecto se pueden generar a partir de cualquier fuente que describa la lógica funcional (es decir, las “reglas”) de un programa, tales como las historias de usuario o los diagramas de flujo. Éstas pueden ser útiles para obtener una visión gráfica de la estructura lógica de un programa y normalmente se utilizan como base para crear tablas de decisión. La captura de decisiones como gráficos causa-efecto y/o tablas de decisión permite alcanzar una cobertura sistemática de la lógica del programa.

Aplicabilidad
Los gráficos causa-efecto son aplicables en las mismas situaciones que las tablas de decisión y también en los mismos niveles de prueba. En particular, un gráfico causa-efecto muestra las combinaciones de condiciones que causan resultados (casualidad), las combinaciones de condiciones que excluyen resultados (NOT[23]), condiciones múltiples que deben ser ciertas para provocar un resultado (AND) y condiciones alternativas que pueden ser ciertas para provocar un resultado específico (OR). Estas relaciones son más fáciles de ver en un gráfico causa-efecto que en una descripción narrativa.

Limitaciones/Dificultades
La representación causa-efecto requiere más tiempo y esfuerzo para aprender que otras técnicas de diseño de pruebas. También requiere el soporte de herramientas. Los gráficos causa-efecto tienen una notación específica que el creador y el lector de los gráficos deben conocer.

Cobertura
Debe probarse cada una de las posibles causas con la línea de efecto, incluyendo la combinación de condiciones, para alcanzar la cobertura mínima. Los gráficos causa-efecto incluyen un medio para definir restricciones en los datos y restricciones en el flujo de lógica.

Tipos de defectos
Estos gráficos detectan los mismos tipos de defectos combinatorios que los detectados con las tablas de decisión. Además, la creación de los gráficos ayudan a definir el nivel de detalle requerido en la base de prueba, y por lo tanto ayuda a mejorar el detalle y la calidad de la base de prueba y ayuda al probador a identificar los requisitos que faltan.

[23] Como operador de negación booleano.

3.2.5 Pruebas de Transición de Estado

Las pruebas de transición de estado se utilizan para probar la capacidad del software para entrar y salir de los distintos estados definidos a través de transiciones válidas y no válidas. Los eventos hacen que el software pase de un estado a otro y lleve a cabo acciones. Los eventos pueden estar cualificados por condiciones (a menudo denominadas condiciones de guarda o guardas de transición) que influyen en el camino de transición a tomar. Por ejemplo, un evento de conexión[24] con una combinación válida de nombre de usuario/contraseña resultará en una transición distinta que un evento de conexión con una contraseña no válida.

El seguimiento de las transiciones de estado se realiza a partir de un diagrama de transiciones que muestra todas las transiciones válidas entre estados en un formato gráfico o en una tabla de estado que recoge todas las posibles transiciones, tanto las válidas como las no válidas.

Aplicabilidad
Las pruebas de transición de estado son aplicables a cualquier software que tenga estados definidos y que tenga eventos que provoquen la transición entre dichos estados (por ejemplo, pantallas cambiantes). Las pruebas de transición de estado pueden utilizarse en cualquier nivel de prueba. El software embebido, aplicaciones web y cualquier tipo de software transaccional son buenos candidatos para este tipo de pruebas. Los sistemas de control, es decir, los controladores de semáforos, también son buenos candidatos para este tipo de pruebas.

Limitaciones/Dificultades
La determinación de los estados suele ser la parte más difícil de la definición de la tabla o el diagrama de estado. Cuando el software tiene una interfaz de usuario, las distintas pantallas que se muestran al usuario, a menudo, se utilizan para definir los estados. Para el software embebido, los estados pueden depender de los estados que experimente el hardware.

Además de los propios estados, la unidad básica de las pruebas de transición de estado es la transición individual, también conocida como conmutador de orden 0[25]. El mero hecho de probar todas las transiciones detectará ciertos tipos de defectos de transición de estado, pero pueden encontrarse más si se prueban secuencias de transacciones. La secuencia de dos transiciones sucesivas se conoce como conmutador de orden 1[26], la secuencia de tres transiciones sucesivas es un conmutador de orden 2, y así sucesivamente. (Estos conmutadores, a veces, se pueden designar alternativamente como N-1 conmutaciones, donde N representa el número de transiciones que se atravesarán). Una única transición, por ejemplo (un conmutador de orden 0), sería una 1-1 conmutación. [Bath08]

Cobertura
Al igual que con otros tipos de técnicas de prueba, existe una jerarquía de niveles de cobertura de las pruebas. El mínimo grado aceptable de cobertura es haber visitado todos los estados y atravesado todas las transiciones. Un 100% de cobertura de transición (también conocida como 100% de cobertura de conmutador de orden 0 o 100% de cobertura de rama lógica) garantizará que se han visitado todos los estados y que se han atravesado todas las transiciones, a menos que el diseño del sistema o el modelo de transición de estado (diagrama o tabla) sean defectuosos. Dependiendo de las relaciones entre estados y transiciones, puede ser necesario atravesar algunas transiciones más de una vez para ejecutar otras transiciones una única vez.

El término “cobertura de conmutador de orden N”[27] se refiere al número de transiciones cubiertas. Por ejemplo, para lograr un 100% de cobertura de conmutador de orden 1[28] es necesario haber probado todas las secuencias válidas de dos transiciones sucesivas por lo menos una vez. Estas pruebas pueden estimular ciertos tipos de fallos que un 100% de cobertura de conmutador de orden 0 omitiría.

La “cobertura de ida y vuelta” se aplica en situaciones en las que las secuencias de transiciones forman bucles. Se alcanza un 100% de cobertura de ida y vuelta cuando se han probado todos los bucles a

[24] Inicio de sesión de usuario.
[25] “0-conmutación”.
[26] 1-conmutación.
[27] cobertura de N-conmutaciones.
[28] 100% de cobertura de 1-conmutación.

partir de un estado que vuelven al mismo estado[29]. Esto debe probarse para todos los estados incluidos en los bucles.

Para cualquiera de estos enfoques, un mayor grado de cobertura intentará incluir todas las transiciones no válidas. Si se incluyen transiciones no válidas en las pruebas de transición de estado, se deben identificar los requisitos de cobertura y los conjuntos de cobertura.

Tipos de defectos
Los tipos de defectos típicos incluyen el procesamiento incorrecto en el estado actual resultante del procesamiento ocurrido en un estado anterior, las transiciones incorrectas o no soportadas, los estados sin salidas y la necesidad de estados y transiciones que no existen. Durante la creación del modelo de máquina de estado, pueden encontrarse defectos en el documento de especificaciones. Los tipos de defectos más comunes son las omisiones (no hay información sobre qué debería ocurrir, en realidad, en una situación dada) y las contradicciones.

3.2.6 Técnicas de Pruebas Combinatorias

Las pruebas combinatorias se utilizan cuando se prueba software con múltiples parámetros, cada uno de ellos con varios valores, lo que da lugar a más combinaciones de las que es posible probar en el tiempo permitido. Los parámetros deben ser independientes y compatibles en el sentido de que cualquier opción de cualquier factor pueda combinarse con cualquier opción de cualquier otro factor. Los árboles de clasificación permiten excluir ciertas combinaciones, si hay determinadas opciones que son incompatibles. Esto no supone que los factores combinados no se afectarán mutuamente, podrían hacerlo perfectamente, pero deberían afectarse mutuamente de una forma aceptable.

Las pruebas combinatorias proporcionan un medio para identificar un subconjunto adecuado de estas combinaciones para alcanzar un nivel de cobertura predeterminado. El Analista de Pruebas puede seleccionar el número de elementos a incluir en las combinaciones, incluyendo elementos individuales, pares, triples o mayores [Copeland03]. Hay una serie de herramientas a disposición del Analista de Pruebas para ayudarle en esta tarea (ver www.pairwise.org para ejemplos). Estas herramientas requieren la representación de los parámetros y sus valores en forma de listado (pruebas de a pares de elementos y pruebas de arreglos ortogonales) o en forma de gráfico (árboles de clasificación) [Grochtmann94]. Las pruebas de a pares de elementos constituyen un método para probar combinaciones de pares de variables de prueba. Los arreglos ortogonales son tablas predefinidas y matemáticamente exactas que permiten al Analista de Pruebas sustituir los elementos a probar por las variables del arreglo, generando un conjunto de combinaciones que alcanzará un nivel de cobertura durante las pruebas [Koomen06]. Las herramientas de árbol de clasificación permiten al Analista de Pruebas definir el tamaño de las combinaciones a probar (es decir, combinaciones de dos valores, tres valores, etc.).

Aplicabilidad
El problema de tener demasiadas combinaciones de valores de parámetros se pone de manifiesto en al menos dos situaciones distintas asociadas a las pruebas. Algunos casos de prueba contienen varios parámetros con varios posibles valores cada uno, por ejemplo una pantalla con varios campos de entrada. En este caso, las combinaciones de los valores de los parámetros constituyen los datos de entrada de los casos de prueba. Además, las dimensiones de algunos sistemas pueden configurarse, lo que da lugar a un espacio de configuración potencialmente amplio. En ambas situaciones, se pueden utilizar pruebas combinatorias para identificar un subconjunto de combinaciones, con un tamaño viable.

En el caso de parámetros con un gran número de valores, antes de ejecutar las pruebas combinatorias para reducir el conjunto de combinaciones resultantes, puede aplicarse la partición de clase de equivalencia, o cualquier otro mecanismo de selección, a cada parámetro individual para reducir el número de valores asociado a cada parámetro.

Estas técnicas normalmente se aplican en los niveles de pruebas de integración, pruebas de sistema y pruebas de integración de sistemas.

[29] al mismo estado de partida.

Limitaciones/Dificultades
La principal limitación que plantean estas técnicas es el supuesto de que los resultados de varias pruebas son representativos de todas las pruebas, y que dichas pocas pruebas representan el uso previsto. Si hubiera una interacción no esperada entre determinadas variables, este tipo de pruebas puede no detectarla si no se prueba esa combinación en particular. Estas técnicas pueden ser difíciles de explicar a un público no técnico, ya que pueden no entender la reducción lógica de las pruebas.

A veces resulta difícil identificar los parámetros y sus valores respectivos. No es fácil encontrar manualmente un conjunto mínimo de combinaciones que cumpla un determinado nivel de cobertura. Por lo general, se utilizan herramientas para identificar el conjunto mínimo de combinaciones. Algunas herramientas tienen la capacidad de forzar la inclusión o exclusión de ciertas (sub)combinaciones en la selección final de combinaciones. El Analista de Pruebas puede emplear esta capacidad para enfatizar o atenuar factores basándose en el conocimiento del dominio o en información sobre el uso del producto.

Cobertura
Existen diversos niveles de cobertura. El nivel más bajo de cobertura se denomina cobertura de a 1 elemento[30] o cobertura de instancia única[31]. Requiere que todos los valores de cada parámetro estén presentes en, al menos una, de las combinaciones elegidas. El siguiente nivel de cobertura se denomina cobertura de a 2 elementos[32] o cobertura de a pares de elementos. Requiere que todos los pares de valores de dos parámetros cualesquiera estén incluidos en al menos una combinación. Esta idea puede generalizarse hasta la cobertura de a n elementos[33], que requiere que todas las subcombinaciones de valores de cualquier conjunto de n parámetros estén incluidas en el conjunto de combinaciones seleccionadas. Cuanto más alto sea el valor de n, más combinaciones serán necesarias para alcanzar la cobertura al 100%. La cobertura mínima con estas técnicas es tener un caso de prueba para cada combinación generada por la herramienta.

Tipos de defectos
El tipo de defecto más común que se detecta con este tipo de prueba son defectos asociados a los valores combinados de varios parámetros.

3.2.7 Pruebas de Caso de Uso

Las pruebas de caso de uso proporcionan pruebas transaccionales, basadas en escenarios, que deberían emular el uso del sistema. Los casos de uso se definen en términos de interacciones entre los actores y el sistema que logran algún objetivo. Los actores pueden ser usuarios o sistemas externos.

Aplicabilidad
Las pruebas de caso de uso normalmente se aplican en los niveles de pruebas de sistema y de aceptación. También pueden utilizarse en las pruebas de integración, en función del nivel de integración, e incluso en las pruebas de componente, en función del comportamiento del componente. Los casos de uso, a menudo, conforman la base para las pruebas de rendimiento porque reflejan un uso realista del sistema. Los escenarios descritos en los casos de uso pueden asignarse a usuarios virtuales para crear una carga realista en el sistema.

Limitaciones/Dificultades
Para ser válidos, los casos de uso deben reflejar transacciones de usuario realistas. Esta información debería proceder de un usuario o de un representante de los usuarios. El valor de un caso de uso se reduce si el caso de uso no refleja con exactitud las actividades del usuario real. Para que la cobertura de pruebas sea rigurosa es importante disponer de una definición exacta de los distintos caminos alternativos (flujos). Los casos de uso deben tomarse como una orientación y no como una definición completa de lo que debería probarse, ya que es posible que no ofrezcan una definición clara de todo el conjunto de requisitos. También puede ser ventajoso crear otros modelos, tales como diagramas de

[30] "cobertura de a 1 elemento" es la traducción de "1-wise coverage".
[31] "cobertura de instancia única" es la traducción de "singleton coverage".
[32] "cobertura de a 2 elementos" es la traducción de "2-wise coverage".
[33] "cobertura de a n elementos" es la traducción de "n-wise coverage".

flujo, a partir de la narrativa de los casos de uso para mejorar la exactitud de las pruebas y comprobar el propio caso de uso.

Cobertura
La cobertura mínima de un caso de uso es tener un caso de prueba para el camino principal (positivo), y un caso de prueba para cada camino alternativo o flujo. Los caminos alternativos incluyen caminos de excepción y fallo. A menudo los caminos alternativos se muestran como extensiones del camino principal. El porcentaje de cobertura se establece dividiendo el número de caminos probados entre el número total de caminos principal y alternativos.

Tipos de defectos
Los defectos incluyen el tratamiento incorrecto de los escenarios definidos, la omisión del tratamiento de caminos alternativos, el procesamiento incorrecto de las condiciones presentadas y la gestión inadecuada o errónea de la información de los errores.

3.2.8 Pruebas de Historias de Usuario

En algunas metodologías Ágiles, como Scrum, los requisitos se preparan en forma de historias de usuario que describen pequeñas unidades funcionales que pueden diseñarse, probarse y demostrarse en una única interacción [Cohn04]. Estas historias de usuario incluyen una descripción de la funcionalidad a implementar, cualesquiera criterios no funcionales, y además incluyen los criterios de aceptación que deben cumplirse para que la historia de usuario pueda considerarse completa.

Aplicabilidad
Las historias de usuario se utilizan principalmente en entornos Ágiles y entornos iterativos e incrementales similares. Se utilizan tanto para pruebas funcionales como para pruebas no funcionales. Las historias de usuario se emplean en todos los niveles, con la previsión de que el desarrollador demuestre la funcionalidad implementada para la historia de usuario antes de entregar el código a los miembros del equipo con el siguiente nivel de tareas de prueba (por ejemplo, pruebas de integración o de rendimiento).

Limitaciones/Dificultades
Dado que las historias son pequeños incrementos de funcionalidad, puede existir el requisito de generar controladores y stubs para probar realmente el tramo de funcionalidad entregada. Normalmente esto exige una capacidad para programar y utilizar herramientas que ayuden a realizar estas pruebas, tales como herramientas de prueba de la API. Normalmente la creación de los controladores y stubs compete al desarrollador, aunque el Analista de Pruebas Técnicas también puede participar en la generación de este código y en el uso de las herramientas de prueba de la API. Si se sigue un modelo de integración continua, como en el caso de la mayoría de proyectos Ágiles, se minimiza la necesidad de controladores y stubs.

Cobertura
La cobertura mínima de una historia de usuario es comprobar que se han cumplido todos los criterios de aceptación especificados.

Tipos de defectos
Normalmente los defectos son funcionales, en el sentido de que el software no ofrece la funcionalidad especificada. También pueden surgir defectos por problemas de integración de la funcionalidad en la nueva historia con la funcionalidad ya existente. Dado que las historias pueden desarrollarse de manera independiente, pueden surgir problemas de rendimiento, interfaz y tratamiento de errores. Es importante que el Analista de Pruebas lleve a cabo tanto pruebas de la funcionalidad individual facilitada como pruebas de integración cada vez que se entregue una nueva historia para su prueba.

3.2.9 Análisis de Dominio

Un dominio es un conjunto definido de valores. El conjunto puede definirse como un rango de valores de una variable única (un dominio unidimensional, por ejemplo, “hombres mayores de 24 y menores de 66”), o como rangos de valores de variables que interactúan (un dominio multidimensional, por ejemplo, “(hombres mayores de 24 y menores de 66) Y (que pesen más de 69 kg y menos de 90 kg)”). Cada

caso de prueba en un dominio multidimensional debe incluir los valores adecuados para cada variable implicada.

El análisis de dominio de un dominio unidimensional normalmente emplea particiones de equivalencia y análisis de valores frontera. Una vez definidas las particiones, el Analista de Pruebas selecciona valores de cada partición que representan un valor interior a la partición (INTERIOR[34]), fuera de la partición (EXTERIOR[35]), sobre la frontera de la partición (SOBRE[36]) y justo después de la frontera de la partición (FUERA[37]). Al establecer estos valores, cada partición se prueba junto con sus condiciones frontera. [Black07]

En el caso de los dominios multidimensionales, el número de casos de prueba generado por estos métodos aumenta de manera exponencial con el número de variables involucradas, mientras que un enfoque basado en la teoría de dominio supone un crecimiento lineal. Asimismo, dado que el enfoque formal incorpora una teoría de defectos (un modelo de defectos[38]), de la que la segmentación de equivalencia y el análisis de valores frontera carecen, su menor conjunto de pruebas encontrará defectos en dominios multidimensionales que el conjunto de pruebas más grande y heurístico seguramente no detectaría. Al tratar con dominios multidimensionales, el modelo de pruebas puede construirse como una tabla de decisión (o “matriz de dominio”). Es probable que para identificar los valores de los casos de prueba en dominios multidimensionales que tienen más de tres dimensiones se requiera soporte informático.

Aplicabilidad
El análisis de dominio combina las técnicas empleadas en las tablas de decisión, las particiones de equivalencia y el análisis de valores frontera para crear un conjunto más pequeño de pruebas que sigan cubriendo las áreas importantes y las áreas de fallo probables. A menudo se aplica en casos en los que las tablas de decisión serían rígidas dado el gran número de posibles variables que interactúan. El análisis de dominio puede llevarse a cabo en cualquier nivel de prueba, pero lo más habitual es que se aplique en los niveles de pruebas de integración y de sistema.

Limitaciones/Dificultades
Realizar un análisis de dominio en profundidad requiere un fuerte conocimiento del software para identificar los distintos dominios y la posible interacción entre ellos. Si un dominio queda sin identificar, las pruebas pueden resultar incompletas, si bien es probable que el dominio llegue a detectarse porque las variables FUERA y EXTERIOR pueden caer en el dominio no detectado. El análisis de dominio constituye una técnica fuerte a aplicar durante el trabajo con un desarrollador para definir las áreas de prueba.

Cobertura
La cobertura mínima para el análisis de dominio es tener una prueba para cada valor INTERIOR, EXTERIOR, SOBRE y FUERA en cada dominio. Cuando los valores se solapan (por ejemplo, el valor EXTERIOR de un dominio es un valor INTERIOR de otro dominio), no hace falta duplicar las pruebas. Gracias a ello, el número de pruebas realmente necesarias a menudo no llega a cuatro por dominio.

Tipos de defectos
Los defectos incluyen problemas funcionales dentro del dominio, el tratamiento de valores límite, problemas de interacción de variables y el tratamiento de errores (en particular para los valores que no están un dominio válido).

3.2.10 Combinación de Técnicas

En ocasiones, las técnicas se combinan para crear casos de prueba. Por ejemplo, las condiciones identificadas en una tabla de decisión podrían someterse a una segmentación de equivalencia para

[34] INTERIOR es la traducción de IN.
[35] EXTERIOR es la traducción de OUT.
[36] SOBRE es la traducción de ON.
[37] FUERA es la traducción de OFF.
[38] “modelo de defectos” es la traducción de “fault model”.

descubrir las múltiples formas con las que cumplir una condición. Los casos de prueba cubrirían entonces no solo cada combinación de condiciones, sino también, para aquellas condiciones que fueran segmentadas, se generarían casos de prueba adicionales para cubrir las particiones de equivalencia. A la hora de seleccionar la técnica específica a aplicar, el Analista de Pruebas debe tener en cuenta la aplicabilidad de la técnica, las limitaciones y dificultades, y los objetivos de las pruebas en términos de cobertura y defectos a detectar. Es posible que no exista una única "mejor" técnica para una situación dada. El uso de técnicas combinadas, a menudo, será lo que aporte la cobertura más completa, asumiendo que se cuenta con el tiempo y las habilidades suficientes para aplicarlas correctamente.

3.3 Técnicas Basadas en Defectos

3.3.1 Uso de Técnicas Basadas en Defectos

Una técnica de diseño de pruebas basada en defectos es aquella en la que el tipo de defecto buscado se utiliza como base para el diseño de la prueba, con pruebas derivadas de forma sistemática de lo que se conoce sobre el tipo de defecto. Al contrario que las pruebas basadas en la especificación que derivan sus pruebas de la especificación, las pruebas basadas en defectos obtienen las pruebas a partir de taxonomías de defectos (es decir, listas categorizadas) que pueden ser completamente independientes del software que se está probando. Las taxonomías pueden incluir listas de tipos de defectos, causas raíz, síntomas de fallo y otros datos asociados a los defectos. Las pruebas basadas en defectos también pueden basarse en listas de riesgos identificados y escenarios de riesgos de las pruebas objetivo. Esta técnica de prueba permite al probador centrarse en un tipo específico de defecto o trabajar sistemáticamente a través de una taxonomía de defectos para defectos conocidos y comunes de un tipo particular. El Analista de Pruebas utiliza los datos de la taxonomía para establecer el objetivo de las pruebas, que es encontrar un tipo de defecto específico. Basándose en esta información, el Analista de Pruebas creará los casos de prueba y las condiciones de prueba que harán que el defecto se manifieste, si existe.

Aplicabilidad
Las pruebas basadas en defectos pueden aplicarse en cualquier nivel de prueba, pero normalmente se utilizan durante las pruebas de sistema. Existen taxonomías normalizadas que son aplicables a múltiples tipos de software. Este tipo de prueba no específica de un producto ayuda a aprovechar el conocimiento estándar del sector[39] para obtener las pruebas particulares. La adhesión a taxonomías específicas de un sector permite realizar un seguimiento de las métricas sobre la ocurrencia de defectos a nivel de proyecto e incluso a nivel de organización.

Limitaciones/Dificultades
Existen múltiples taxonomías de defectos y pueden centrarse en tipos de pruebas específicos, tales como las pruebas de usabilidad. Es importante escoger una taxonomía que sea aplicable al software que se está probando, si hay alguna disponible. Por ejemplo, es posible que no haya ninguna taxonomía disponible para un software innovador. Algunas organizaciones han compilado sus propias taxonomías de defectos probables o frecuentes. Independientemente de la taxonomía empleada, es importante definir la cobertura esperada antes de iniciar las pruebas.

Cobertura
La técnica aporta criterios de cobertura que sirven para determinar cuándo se han identificado todos los casos de prueba útiles. Como cuestión práctica, los criterios de cobertura para las técnicas basadas en los defectos tienden a ser menos sistemáticos que para las técnicas basadas en la especificación, en el sentido de que solo se dan reglas generales para la cobertura y que la decisión específica sobre lo que constituye el límite de la cobertura útil es discrecional. Al igual que otras técnicas, los criterios de cobertura no implican que el conjunto de pruebas esté completo, sino más bien que los defectos considerados ya no sugieren ninguna prueba útil basada en esa técnica.

Tipos de defectos
Los tipos de defectos descubiertos normalmente dependen de la taxonomía que se utilice. Si se utiliza una taxonomía de interfaz de usuario, la mayoría de los defectos detectados probablemente se refieran

[39] "sector" es la traducción de "industry".

a la interfaz de usuario, pero también pueden detectarse otros defectos como un subproductos de las pruebas específicas.

3.3.2 Taxonomías de Defectos

Las taxonomías de defectos son listas categorizadas de tipos de defectos. Estas listas pueden ser muy generales y utilizarse como orientaciones de alto nivel o pueden ser muy específicas. Por ejemplo, una taxonomía para defectos de interfaz de usuario podría contener elementos generales tales como funcionalidad, tratamiento de errores, presentación de gráficos y rendimiento. Una taxonomía detallada podría incluir una lista de todos los objetos posibles de interfaz de usuario (especialmente para una interfaz gráfica de usuario) y podría indicar el tratamiento inadecuado de estos objetos, tales como:

- Campo de texto
  - no se aceptan datos válidos.
  - se aceptan datos no válidos.
  - no se comprueba la longitud de la entrada.
  - no se detectan caracteres especiales.
  - los mensajes de error de usuario no son informativos.
  - el usuario no puede corregir datos erróneos.
  - no se aplican las reglas.
- Campo de fecha
  - no se aceptan fechas válidas.
  - no se rechazan las fechas no válidas.
  - no se comprueban los rangos de fechas.
  - los datos de precisión no se tratan correctamente (por ejemplo, hh:mm:ss).
  - el usuario no puede corregir datos erróneos.
  - no se aplican las reglas (por ejemplo, la fecha de fin debe ser posterior a la fecha de inicio).

Hay una gran variedad de taxonomías de defectos disponibles, desde taxonomías formales que pueden comprarse hasta taxonomías diseñadas con fines específicos por distintas organizaciones. Las taxonomías desarrolladas internamente también pueden utilizarse para abordar defectos específicos que se dan con frecuencia dentro de la organización. A la hora de crear una nueva taxonomía de defectos o de personalizar uno disponible, es importante definir primero los objetivos o metas de la taxonomía. Por ejemplo, el objetivo puede ser identificar problemas de interfaz de usuario detectados en sistemas de producción o identificar problemas relacionados con el tratamiento de los campos de entrada.

Para crear una taxonomía:

1. Crear un objetivo y definir el nivel de detalle deseado.
2. Seleccionar una taxonomía dada para utilizarla como base.
3. Definir los valores y defectos comunes vividos en la organización y/o a partir de prácticas en el exterior.

Cuanto más detallada sea la taxonomía, más tiempo llevará su desarrollo y mantenimiento, pero mayor será el nivel de reproducibilidad de los resultados de las pruebas. Las taxonomías detalladas pueden ser redundantes, pero permiten al equipo de prueba dividir las pruebas sin pérdida de información o cobertura.

Una vez seleccionada la taxonomía adecuada, puede utilizarse para crear condiciones de prueba y casos de prueba. Una taxonomía basada en riesgos puede ayudar a concentrar las pruebas en un área de riesgo específica. Las taxonomías también pueden utilizarse para áreas no funcionales tales como usabilidad, rendimiento, etc. Hay listas de taxonomías disponibles en varias publicaciones, de IEEE[40] y en Internet.

[40] "IEEE" es el acrónimo de "Institute of Electrical and Electronics Engineers".

3.4 Técnicas Basadas en la Experiencia

Las pruebas basadas en la experiencia utilizan las capacidades e intuición de los probadores, junto con su experiencia en aplicaciones o tecnologías similares. Estas pruebas son efectivas en la detección de defectos, pero no son tan adecuadas como otras técnicas para lograr niveles de cobertura de prueba específicos o generar procedimientos de prueba reutilizables. En los casos en los que la documentación del sistema es insuficiente, el tiempo de prueba está severamente restringido o el equipo de pruebas tiene mucha experiencia en el sistema a probar, las pruebas basadas en la experiencia pueden constituir una buena alternativa a enfoques más estructurados. Las pruebas basadas en la experiencia pueden no ser adecuadas en sistemas que requieren una documentación de prueba detallada, altos niveles de repetibilidad o una capacidad para evaluar con precisión la cobertura de las pruebas.

Cuando se utilizan enfoques dinámicos y heurísticos, los probadores normalmente utilizan pruebas basadas en la experiencia, y las pruebas son más reactivas a los acontecimientos que los enfoques de prueba con una planificación previa. Además, la ejecución y la evaluación son tareas concurrentes. Algunos enfoques estructurados para las pruebas basadas en la experiencia no son totalmente dinámicos, es decir, las pruebas no se crean, en su totalidad, en el mismo momento en el que el probador ejecuta la prueba.

Cabe destacar que si bien se presentan algunas ideas sobre cobertura para las técnicas aquí abordadas, las técnicas basadas en la experiencia no tienen criterios de cobertura formales.

3.4.1 Predicción de Errores

Al utilizar la técnica de predicción de errores, el Analista de Pruebas usa su experiencia para conjeturar posibles errores que pueden haberse cometido durante el diseño y desarrollo del código. Una vez identificados los errores esperados, el Analista de Pruebas determina los mejores métodos a emplear para descubrir los defectos resultantes. Por ejemplo, si el Analista de Pruebas espera que el software presente fallos cuando se introduce una contraseña, las pruebas se diseñarán para introducir una serie de valores distintos en el campo contraseña para comprobar si efectivamente se ha producido el error y ha resultado en un defecto que puede verse como un fallo al ejecutar las pruebas.

Además de utilizarse como técnica de prueba, la predicción de errores también es útil durante el análisis de riesgos a la hora de identificar modos de fallo potenciales. [Myers79]

Aplicabilidad
La predicción de errores se realiza principalmente durante las pruebas de integración y sistema, pero se pueden utilizar en cualquier nivel de prueba. Esta técnica se utiliza, a menudo, con otras técnicas y ayuda a ampliar el alcance de los casos de prueba existentes. La predicción de errores también puede utilizarse de manera efectiva durante las pruebas de una nueva entrega del software para probar errores comunes y errores antes de iniciar pruebas[41] más rigurosas y mediante guion. Las listas de comprobación y las taxonomías pueden ser de utilidad a la hora de orientar las pruebas.

Limitaciones/Dificultades
La cobertura es difícil de evaluar y varía mucho en función de la capacidad y la experiencia del Analista de Pruebas. El probador experimentado que conozca los tipos de defectos que normalmente se introducen en el tipo de código sometido a pruebas es la persona idónea para utilizar esta técnica. La predicción de errores se utiliza con mucha frecuencia, pero muchas veces no está documentada y, por lo tanto, puede ser menos reproducible que otras formas de prueba.

Cobertura
Cuando se utiliza una taxonomía, la cobertura viene determinada por los correspondientes defectos de datos y los tipos de defectos. Sin una taxonomía, la cobertura se ve limitada por la experiencia y el conocimiento del probador y por el tiempo disponible. El resultado de esta técnica variará en función de la habilidad con la que el probador aborda las distintas áreas problemáticas.

Tipos de defectos

[41] "pruebas mediante guion" (antes "pruebas programadas") es la traducción de "scripted testing".

Los defectos típicos suelen coincidir con los definidos en la taxonomía particular o "predicha"[42] por el Analista de Pruebas, que puede que no se hayan detectado mediante pruebas basadas en la especificación.

3.4.2 Pruebas Basadas en Listas de Comprobación

Al aplicar la técnica de pruebas basadas en listas de comprobación, un Analista de Pruebas con experiencia utiliza una lista generalizada de elementos de alto nivel que se deben anotar, revisar o recordar o un conjunto de normas o criterios respecto de los cuales se debe verificar un producto. Estas listas de comprobación se crean tomando como base una serie de normas, experiencia y otras consideraciones. Un ejemplo de pruebas basadas en listas de comprobación es una lista de comprobación de estándares de interfaz de usuario utilizada como la base para las pruebas de una aplicación.

Aplicabilidad
Las pruebas basadas en listas de comprobación son especialmente efectivas en proyectos con un equipo de pruebas experimentado que conozca el software sujeto a prueba o que esté familiarizado con el área cubierta por la lista de comprobación (por ejemplo, para aplicar con éxito una lista de comprobación de interfaz de usuario, el Analista de Pruebas puede estar familiarizado con las pruebas de interfaz de usuario pero no con el software específico sujeto a prueba). Dado que las listas de comprobación son de alto nivel y tienden a no incluir los pasos detallados que suelen encontrarse en los casos de prueba y en los procedimientos de prueba, se utiliza el conocimiento del probador para cubrir esos vacíos. Al eliminar los pasos detallados, las listas de comprobación requieren poco mantenimiento y pueden aplicarse a varias entregas similares. Las listas de comprobación pueden utilizarse en cualquier nivel de pruebas. Las listas de comprobación también se utilizan en pruebas de regresión y pruebas de humo.

Limitaciones/Dificultades
La naturaleza de alto nivel de las listas de comprobación puede afectar a la reproducibilidad de los resultados de las pruebas. Es posible que distintos probadores interpreten las listas de comprobación de forma diferente y sigan distintos enfoques para cumplir los elementos de la lista de comprobación. Esto puede ser la causa de la obtención de distintos resultados, a pesar de utilizar la misma lista de comprobación. Esto puede dar lugar a una mayor cobertura, pero a veces a costa de la reproducibilidad. Las listas de comprobación también pueden dar lugar a un exceso de confianza en lo que respecta al nivel de cobertura alcanzado, dado que las pruebas reales dependen del criterio del probador. Se pueden obtener listas de comprobación a partir de casos de prueba o de listas con más detalles y tienden a crecer con el tiempo. Es necesario cierto mantenimiento para asegurar que las listas de comprobación cubren los aspectos más importantes del software sujeto a prueba.

Cobertura
La cobertura es tan buena como la de la lista de comprobación, dado el carácter de alto nivel de la lista de comprobación, los resultados variarán en función del Analista de Pruebas que ejecute la lista de comprobación.

Tipos de defectos
Los defectos típicos que esta técnica detecta incluyen fallos derivados de la modificación de los datos, la secuencia de pasos o el flujo de trabajo general durante las pruebas. El uso de listas de comprobación permite que las pruebas se mantengan actualizadas gracias a que pueden incluirse nuevas combinaciones de datos y procesos durante las pruebas.

3.4.3 Pruebas Exploratorias

Las pruebas exploratorias se caracterizan porque el probador, de forma simultánea, aprende sobre el producto y sobre sus defectos, planifica el trabajo de pruebas a realizar, diseña y ejecuta las pruebas e informa sobre los resultados. El probador ajusta de forma dinámica los objetivos de la prueba durante la fase de ejecución y solo elabora documentación ligera[43]. [Whittaker09]

[42] Anticipada
[43] "ligera" es la traducción de "lightweight".

Aplicabilidad
Las buenas pruebas exploratorias están planificadas, son interactivas y creativas. Esta técnica requiere escasa documentación sobre el sistema a probar y suele utilizarse en situaciones en las que no hay documentación disponible o adecuada para otras técnicas de prueba. Las pruebas exploratorias se utilizan a menudo para ampliar otras pruebas y sirven de base para el desarrollo de casos de prueba adicionales.

Limitaciones/Dificultades
Las pruebas exploratorias pueden ser difíciles de gestionar y programar[44]. La cobertura puede ser esporádica y la reproducibilidad es difícil. Un método utilizado para gestionar pruebas exploratorias se basa en el uso de contratos[45] para designar qué áreas deben cubrirse en una sesión de pruebas y duración temporal preestablecida para determinar el tiempo que debe dedicarse a las pruebas. Al término de una sesión de pruebas o conjunto de sesiones, el Jefe de Pruebas puede celebrar una sesión de recapitulación para recopilar los resultados de las pruebas y establecer los contratos para las próximas sesiones. Las sesiones de recapitulación son difíciles de escalar[46] en equipos de prueba o proyectos grandes.

Otra dificultad que plantean las sesiones exploratorias es realizar su seguimiento con exactitud en un sistema de gestión de pruebas. A veces para ello se crean casos de prueba que en realidad son sesiones exploratorias. Esto permite hacer el seguimiento del tiempo asignado a las pruebas exploratorias y de la cobertura prevista junto a los demás esfuerzos de prueba.

Dado que la reproducibilidad con las pruebas exploratorias puede resultar complicada, esto también puede provocar problemas cuando deban repetirse los pasos para reproducir un fallo. Algunas organizaciones usan la funcionalidad de captura/reproducción de una herramienta de automatización de pruebas para registrar los pasos ejecutados por un probador de pruebas exploratorias. Con esto se obtiene un registro completo de todas las actividades durante la sesión exploratoria (o cualquier sesión de pruebas basada en la experiencia). Explorar los detalles para encontrar la causa real del fallo puede resultar tedioso, pero al menos hay un registro de todos los pasos implicados.

Cobertura
Los contratos pueden crearse para especificar tareas, objetivos y entregables. A continuación se planifican las sesiones exploratorias para alcanzar dichos objetivos. El contrato también puede identificar dónde concentrar el esfuerzo de prueba, qué está dentro y fuera del alcance de la sesión de prueba y qué recursos deberían comprometerse para llevar a cabo las pruebas previstas. Una sesión puede utilizarse para concentrarse en tipos específicos de defectos y otra en áreas potencialmente problemáticas que pueden abordarse sin la formalidad de pruebas mediante guion[47] (antes pruebas programadas).

Tipos de defectos
Los defectos típicos que detectan las pruebas exploratorias suelen ser problemas basados en escenarios que han sido obviados durante las pruebas funcionales basadas guion, problemas que se dan entre las fronteras funcionales, y problemas relacionados con el flujo de trabajo. A veces las pruebas exploratorias también revelan problemas de rendimiento y seguridad.

3.4.4 Aplicación de la Mejor Técnica

Las técnicas basadas en defectos y en la experiencia requieren la aplicación de conocimiento sobre defectos y otras experiencias en pruebas para orientar las pruebas con el fin de incrementar la detección de defectos. Éstas varían desde “pruebas rápidas” en las que el probador no tiene que realizar ninguna actividad planificada de forma previa y formal, pasando por sesiones planificadas con antelación hasta sesiones basadas en guion. Casi siempre son útiles pero tienen un valor especial en las siguientes circunstancias:

[44] “programar” es la traducción de “to schedule”.
[45] “contrato de prueba” es la traducción de “test charter”.
[46] dimensionar.
[47] “pruebas mediante guion” es la traducción de “scripted testing”.

- No hay especificaciones disponibles.
- Existe escasa documentación del sistema sujeto a prueba.
- No hay tiempo suficiente para diseñar y crear pruebas detalladas.
- Los probadores tienen experiencia en el campo y/o la tecnología.
- La diversidad respecto de las pruebas mediante guion (antes pruebas programadas) constituye un objetivo para maximizar la cobertura de las pruebas.
- Se deben analizar fallos operativos.

Las técnicas basadas en los defectos y en la experiencia también son útiles cuando se utilizan en combinación con las técnicas basadas en la especificación, ya que salvan las deficiencias de cobertura de prueba resultantes de los puntos débiles sistemáticos de estas técnicas. Al igual que sucede con las técnicas basadas en la especificación, no existe una única técnica perfecta para todas las situaciones. Es importante que el Analista de Pruebas conozca las ventajas y desventajas de cada técnica y que sea capaz de seleccionar la mejor técnica o conjunto de técnicas para cada situación, teniendo en cuenta el tipo de proyecto, el calendario, el acceso a la información, las competencias del probador y otros factores que pueden afectar a la selección.

4. Pruebas de las Características de Calidad del Software - 120 minutos

Palabras Clave

pruebas de accesibilidad, pruebas de exactitud, atractivo, evaluación heurística, pruebas de interoperabilidad, capacidad de ser aprendido[48], operabilidad, pruebas de adecuación, IMUS[49], comprensibilidad[50], pruebas de usabilidad, IAMSW[51].

Objetivos de Aprendizaje para las Pruebas de las Características de Calidad del software

4.2 Características de Calidad para Pruebas de Dominio de Negocio

AP-4.2.1 (K2) Explicar mediante ejemplos qué técnicas de prueba son adecuadas para probar la exactitud, la adecuación, la interoperabilidad y las características de cumplimiento.

AP-4.2.2 (K2) En lo referente a las características de exactitud, adecuación e interoperabilidad, definir los defectos típicos a que se deben fijar como objetivo.

AP-4.2.3 (K2) En cuanto a las características de exactitud, adecuación e interoperabilidad, definir cuándo se debería probar la característica en el ciclo de vida.

AP-4.2.4 (K4) En un contexto de proyecto dado, indicar los enfoques que serían apropiados para verificar y validar tanto la implementación de los requisitos de usabilidad como el cumplimiento de las expectativas del usuario.

[48] "capacidad de ser aprendido" antes "aprendibilidad".
[49] "IMUS" es la traducción de "SUMI".
[50] "capacidad de ser entendido" antes "comprensibilidad".
[51] "IAMSW" es la traducción de "WAMMI".

4.1 Introducción

Mientras que el capítulo anterior describía las técnicas específicas disponibles para el probador, este capítulo abordará cómo se aplican dichas técnicas para evaluar las principales características que se emplean para describir la calidad de los sistemas o de las aplicaciones software.

Este programa de estudio aborda las características de calidad que podría evaluar el Analista de Pruebas . Los atributos que debe evaluar por el Analista de Pruebas Técnicas se abordan en el programa de estudio de "Probador Certificado – Nivel Avanzado - Analista de Pruebas Técnicas". La descripción de las características de calidad de producto aportada por la norma ISO 9126 se utiliza como una guía para describir las características. También se pueden utilizar otros estándares, tales como la serie ISO 25000 [ISO25000] (que reemplaza a la ISO 9126). Las características de calidad ISO se dividen en características de calidad de producto (atributos), cada una de las cuales puede constar a su vez de subcaracterísticas (subatributos). La tabla a continuación muestra estas características e indica qué características/subcaracterísticas deben cubrir los programas de estudio del Analista de Pruebas y del Analista de Pruebas Técnicas:

Característica	Subcaracterística	Analista de pruebas	Analista de Pruebas Técnicas
Funcionalidad	Exactitud, adecuación, interoperabilidad, cumplimiento	x	
	Seguridad		x
Fiabilidad	Madurez (robustez), tolerancia a faltas, capacidad de recuperación[52], cumplimiento		x
Usabilidad	Capacidad de ser entendido[53], capacidad de ser aprendido[54], operabilidad, atractivo, cumplimiento	x	
Eficiencia	Rendimiento (comportamiento temporal), utilización de recursos, cumplimiento		x
Mantenibilidad	Capacidad de ser analizado[55], capacidad para ser modificado[56], estabilidad, capacidad de ser probado[57], cumplimiento		x
Portabilidad	Adaptabilidad, instabilidad, coexistencia, capacidad de ser reemplazado[58], cumplimiento		x

El Analista de Pruebas debería concentrarse en las características de calidad del software de funcionalidad y usabilidad. El Analista de Pruebas también debería llevar a cabo las pruebas de accesibilidad. Si bien la accesibilidad no figura en la lista de subcaracterísticas, a menudo se considera que forma parte de las pruebas de usabilidad. A menudo se considera que las pruebas de las demás características de calidad son competencia del Analista de Pruebas Técnicas. Mientras que esta asignación de trabajo puede variar según las distintas organizaciones, es la seguida por los programas de estudio del ISTQB.

La subcaracterística de cumplimiento está presente en cada característica de calidad. En el caso de algunos entornos de seguridad crítica o regulados, cada característica de calidad puede tener que ajustarse a ciertos estándares y regulaciones específicos (por ejemplo, el cumplimiento de funcionalidad puede indicar que la funcionalidad cumple con un estándar específico, como utilizar un protocolo de comunicación específico, para poder enviar/recibir datos desde un circuito integrado).

[52] "capacidad de recuperación" antes "recuperabilidad".
[53] "capacidad de ser entendido" antes "comprensibilidad".
[54] "capacidad de ser aprendido" antes "aprendibilidad".
[55] "capacidad de ser analizado" antes "analizabilidad".
[56] "capacidad para ser modificado" antes "modificabilidad".
[57] "capacidad de ser probado" antes "testabilidad".
[58] "capacidad de ser reemplazado" antes "reemplazabilidad".

Dado que estos estándares pueden variar mucho en función del sector, no se abordarán en detalle en esta sección. Si el Analista de Pruebas está trabajando en un entorno afectado por requisitos de cumplimiento, es importante comprender dichos requisitos y asegurar que tanto las pruebas como la documentación de las pruebas se ajustarán a los requisitos de cumplimiento.

Para todas las características y subcaracterísticas de calidad estudiadas en esta sección, se deben reconocer los riesgos típicos para poder dar forma y documentar una estrategia de pruebas adecuada. Las pruebas de características de calidad requieren prestar una atención especial al momento adecuado del ciclo de vida, las herramientas necesarias, la disponibilidad de software y documentación y la competencia técnica. Sin la planificación de una estrategia para abordar cada característica y sus necesidades de prueba específicas, el probador puede no disponer de una planificación, de un tiempo de intensificación y ejecución de la prueba adecuados incluidos en el calendario. Algunas de estas pruebas, por ejemplo, las pruebas de usabilidad, pueden requerir la asignación de recursos humanos especiales, planificación extensiva, laboratorios dedicados, habilidades de prueba especializadas y, en la mayoría de los casos, una cantidad importante de tiempo. En algunos casos, las pruebas de usabilidad pueden ser ejecutadas por un grupo experto de usabilidad, o experiencia de usuario, independiente.

Las pruebas de las características y subcaracterísticas de calidad deben estar integradas en el calendario general de pruebas, asignando los recursos adecuados al esfuerzo. Cada una de estas áreas tiene necesidades específicas, problemas específicos de orientación y pueden darse en distintos tiempos durante el ciclo de vida de desarrollo de software, según lo establecido en las siguientes secciones.

Si bien el Analista de Pruebas puede no ser responsable de las características de calidad que requieren un enfoque más técnico, es importante que el Analista de Pruebas conozca el resto de características y entienda las áreas de solapamiento de las pruebas. Así por ejemplo, sin un producto falla en pruebas de rendimiento también es probable que falle en pruebas de usabilidad si es demasiado lento para un uso efectivo para el usuario. De la misma manera, un producto con problemas de interoperabilidad con algunos componentes no esté listo para las pruebas de portabilidad, dado que ello tenderá a ocultar los problemas más básicos cuando se cambie el entorno.

4.2 Características de Calidad para Pruebas de Dominio de Negocio

Las pruebas funcionales constituyen un foco de atención fundamental para el Analista de Pruebas. Las pruebas funcionales se centran en “qué” hace el producto. La base de prueba para las pruebas funcionales es, en general, un documento requisitos o especificación, competencia específica del dominio o la necesidad implícita. Las pruebas funcionales varían en función del nivel de prueba en el que se realizan y también pueden verse afectadas por el ciclo de vida de desarrollo del software. Por ejemplo, una prueba funcional que se lleve a cabo durante las pruebas de integración, probará la funcionalidad de los módulos interconectados[59] que implementan una única función definida. En el nivel de pruebas de sistema, las pruebas funcionales probarán la funcionalidad de la aplicación completa. Para los sistemas de sistemas, las pruebas funcionales se centrarán fundamentalmente en las pruebas de extremo a extremo de los sistemas integrados. En un entorno Ágil, las pruebas funcionales normalmente se limitan a la funcionalidad puesta a disposición en la iteración o sprint particular, aunque las pruebas de regresión de una iteración puedan cubrir toda la funcionalidad entregada.

Durante las pruebas funcionales (ver Capítulo 3) se emplea una gran variedad de técnicas de prueba. Podrá llevar a cabo pruebas funcionales un probador dedicado, un experto del dominio, o un desarrollador (normalmente en el nivel de componente).

Además de las pruebas funcionales cubiertas en esta sección, también hay dos características de calidad que forman parte del área de responsabilidad del Analista de Pruebas y que se consideran áreas de prueba no funcionales (centradas en “cómo” el producto entrega la funcionalidad). Estos dos atributos no funcionales son la usabilidad y la accesibilidad.

[59] “módulos interconectados” es la traducción de “interfacing modules”.

En esta sección se consideran las siguientes características de calidad:

- Subcaracterísticas de calidad funcionales
  - Exactitud
  - Adecuación
  - Interoperabilidad
- Características de calidad no funcionales
  - Usabilidad
  - Accesibilidad

4.2.1 Pruebas de Exactitud

La exactitud funcional comprende el cumplimiento por parte de la aplicación de los requisitos especificados o implícitos y también puede abarcar la exactitud de cálculo. Las pruebas de exactitud emplean muchas de las técnicas de prueba explicadas en el Capítulo 3 y, a menudo, utilizan como oráculo de prueba la especificación o un sistema legado. Las pruebas de exactitud pueden realizarse en cualquier etapa del ciclo de vida y están orientadas al tratamiento incorrecto de datos o situaciones.

4.2.2 Pruebas de Adecuación

Las pruebas de adecuación implican evaluar y validar la idoneidad de un conjunto de funciones para la consecución de las tareas especificadas previstas. Estas pruebas pueden basarse en casos de uso. Las pruebas de adecuación normalmente se realizan durante las pruebas de sistema, pero también pueden realizarse en etapas posteriores de las pruebas de integración. Los defectos descubiertos en estas pruebas son indicaciones de que el sistema no será capaz de satisfacer las necesidades del usuario de una forma que sea considerada aceptable.

4.2.3 Pruebas de Interoperabilidad

Las pruebas de interoperabilidad prueban el grado en el que dos o más sistemas o componentes pueden intercambiar información y, posteriormente, utilizar la información intercambiada. Las pruebas deben cubrir todos los entornos objetivo previstos (incluyendo variaciones en el hardware, software, middleware, sistema operativo, etc.) para asegurar que el intercambio de datos funcionará correctamente. En realidad, esto solo puede ser viable para un número relativamente pequeño de entornos. En ese caso las pruebas de interoperabilidad pueden limitarse a un pequeño grupo representativo de entornos selecto. La especificación de pruebas de interoperabilidad requiere que se identifiquen y configuren las combinaciones de los entornos destino previstos y que estén disponibles para el equipo de pruebas. Entonces estos entornos se prueban empleando una selección de casos de prueba funcionales que se practican en los diferentes puntos de intercambio de datos presentes en el entorno.

La interoperabilidad se refiere a la forma en que diferentes sistemas software interactúan entre sí. Un producto software que presente buenas características interoperabilidad podrá integrarse en otros sistemas sin requerir grandes cambios. El número de cambios y el esfuerzo empleado para realizar tales cambios puede utilizarse como una medida de la interoperabilidad.

Las pruebas de interoperabilidad del software pueden, por ejemplo, centrarse en las siguientes prestaciones de diseño:

- Uso de los estándares de comunicación a nivel industrial, como XML.
- Habilidad para detectar automáticamente las necesidades de comunicación de los sistemas con los que interactúa y realizar los ajustes correspondientes.

Las pruebas de interoperabilidad pueden ser especialmente importantes para las organizaciones que desarrollan software y herramientas comerciales de distribución masiva[60] (“COTS”[61]) y para organizaciones que desarrollan sistemas de sistemas.

[60] “software comercial de distribución masiva” es la traducción de “Commercial Off-The-Shelf software”.
[61] “COTS” es el acrónimo del término en inglés “Commercial Of The Shelf”.

Este tipo de pruebas se lleva a cabo durante las pruebas de integración de componentes y las pruebas de sistema, y se centran en la interacción del sistema con su entorno. A nivel de integración del sistemas, este tipo de pruebas se realiza para establecer lo bien que el sistema totalmente desarrollado interactúa con otros sistemas. Dado que los sistemas pueden interoperar a varios niveles, el Analista de Pruebas debe comprender estas interacciones y ser capaz de crear las condiciones que ejercerán las distintas interacciones. Por ejemplo, si dos sistemas intercambiarán datos, el Analista de Pruebas debe ser capaz de generar los datos necesarios y las transacciones requeridas para llevar a cabo dicho intercambio de datos. Es importante recordar que es posible que no todas las interacciones estén claramente especificadas en los documentos de requisitos. En cambio, muchas de estas interacciones estarán definidas exclusivamente en los documentos de arquitectura y diseño del sistema. El Analista de Pruebas debe estar en condiciones de estudiar esos documentos para establecer los puntos de intercambio de información entre los sistemas y entre el sistema y su entorno con el fin de asegurar que todos son probados. En las pruebas de interoperabilidad pueden aplicarse técnicas como la de tablas de decisión, diagramas de transición de estado, casos de uso y pruebas combinatorias. Entre los defectos típicos que se detectan se incluye el intercambio incorrecto de datos entre los componentes que interactúan.

4.2.4 Pruebas de Usabilidad

Es importante comprender por qué los usuarios pueden tener dificultades al utilizar el sistema. Para poder comprender esta situación, es necesario partir de la base de que el término “usuario” abarca una gran variedad de tipos de personas diferentes, desde expertos en TI, hasta niños o personas con discapacidades.

Algunas instituciones nacionales (por ejemplo, el British Royal National Institute for the Blind) recomiendan que las páginas Web sean también accesibles para usuarios discapacitados, personas invidentes o parcialmente invidentes, con movilidad reducida, sordos o con problemas de aprendizaje. Comprobando que estos sitios Web son usables para los usuarios anteriores también podría mejorar la usabilidad para el resto de usuarios. La accesibilidad se trata más adelante.

Las pruebas de usabilidad prueban la facilidad con la que los usuarios pueden utilizar o aprender a utilizar el sistema para lograr un objetivo específico en un contexto dado. Las pruebas de usabilidad tienen por objeto medir lo siguiente:

- Eficacia - capacidad del producto software para permitir a los usuarios alcanzar objetivos especificados con exactitud y completitud en un contexto de uso especificado.
- Eficiencia - capacidad del producto para permitir que los usuarios emplear cantidades adecuadas de recursos en relación con la efectividad alcanzada en un contexto de uso especificado.
- Satisfacción - capacidad del software para satisfacer a los usuarios en un contexto de uso especificado.

Entre los atributos que pueden ser medidos se encuentran:

- Capacidad de ser entendido - atributos del software que afectan al esfuerzo que necesita invertir el usuario para reconocer el concepto lógico y su aplicabilidad.
- Capacidad de ser aprendido - atributos del software que afectan al esfuerzo que necesita invertir el usuario para aprender el funcionamiento de la aplicación.
- Operabilidad - atributos del software que afectan al esfuerzo que necesita invertir el usuario para llevar a cabo tareas de forma eficaz y eficiente.
- Atractivo - capacidad del software para resultar atractivo al usuario.

Las pruebas de usabilidad por lo general constan de dos pasos:

- Pruebas de Usabilidad Formativa - pruebas que se llevan a cabo de manera iterativa durante las fases de diseño y prototipado para ayudar a guiar (o “formar”) el diseño mediante la identificación de defectos de diseño de usabilidad.
- Pruebas de Usabilidad Sumativa - pruebas que se realizan después de la implementación para medir la usabilidad e identificar problemas con un componente o sistema finalizado.

Entre las habilidades de los probadores de usabilidad deben estar la competencia técnica y la experiencia en las siguientes áreas:

- Sociología.
- Psicología.
- Cumplimiento de estándares nacionales (incluidos los estándares de accesibilidad).
- Ergonomía.

4.2.4.1 Ejecución de Pruebas de Usabilidad

La validación de la implementación actual debe llevarse a cabo en unas condiciones lo más parecidas posibles a aquellas en las que se usará el sistema. Esto puede suponer la instalación de un laboratorio de usabilidad con cámaras de vídeo, maquetas de las oficinas, paneles de revisión, usuarios, etc., para que el personal de desarrollo pueda observar los efectos que tiene un sistema real sobre gente real. Las pruebas de usabilidad formales, a menudo, requieren cierta preparación de los "usuarios" (éstos pueden ser usuarios reales o representantes de usuarios) mediante la entrega de guiones o de instrucciones a seguir. Otras formas de pruebas libres permiten al usuario experimentar con el software de forma que los observadores puedan determinar la facilidad o dificultad con la que el usuario averigua cómo llevar a cabo sus tareas.

El Analista de Pruebas puede ejecutar muchas de las pruebas de usabilidad como parte de otras pruebas, por ejemplo, durante las pruebas de funcionales de sistema. Las directrices de usabilidad pueden ser útiles para la consecución de un enfoque consistente para la detección y comunicación de defectos de usabilidad en todas las fases del ciclo de vida. Sin directrices de usabilidad, puede ser difícil establecer qué es una usabilidad "inaceptable". Por ejemplo, ¿es poco razonable que un usuario tenga que hacer 10 clics con el ratón para acceder a una aplicación? En ausencia de directrices específicas, el Analista de Pruebas puede verse en la difícil situación de tener que defender informes de defectos que el desarrollador quiere cerrar porque el software funciona "según el diseño". Es muy importante contar con especificaciones de usabilidad verificables definidas en los requisitos, además de disponer de una serie de directrices de usabilidad que se apliquen a proyectos similares. Las directrices deberían incluir elementos tales como la accesibilidad de las instrucciones, la claridad de los mensajes, el número de clics necesario para llevar a cabo una actividad, los mensajes de error, los indicadores de procesamiento (algún tipo de indicador que informa al usuario de que el sistema está procesando y no puede aceptar más entradas por el momento), directrices sobre el diseño de la pantalla, el uso de colores y sonidos y otros factores que afectan a la experiencia del usuario.

4.2.4.2 Especificación de Pruebas de Usabilidad

Las principales técnicas que se emplean para las pruebas de usabilidad son:

- Inspección, evaluación o revisión.
- Interacción dinámica con prototipos.
- Verificación y validación de la implementación real.
- Realización de encuestas y cuestionarios

Inspección, evaluación o revisión

La inspección o revisión de la especificación de requisitos y diseños desde la perspectiva de la usabilidad aumenta el nivel de implicación del usuario y puede ser rentable al detectar los problemas de forma temprana. La evaluación heurística (inspección sistemática del diseño de la interfaz de usuario para la usabilidad) se puede aplicar para localizar los problemas de usabilidad en el diseño, para que estos puedan solucionarse como parte de un proceso de diseño iterativo. Esto implica contar con un pequeño grupo de evaluadores que examine la interfaz y juzgue el cumplimiento con principios de usabilidad reconocidos ("heurística"). Las revisiones son más efectivas cuando la interfaz de usuario es más visible. Por ejemplo, muestras de capturas de pantalla suelen ser más fáciles de entender e interpretar que la descripción narrativa de una funcionalidad asociada a una pantalla en particular. La visualización es importante para una revisión de usabilidad adecuada de la documentación.

Interacción de forma dinámica con prototipos

Cuando se desarrollan prototipos, el Analista de Pruebas debe trabajar con los prototipos y ayudar a los desarrolladores a evolucionar el prototipo incorporando al diseño la realimentación[62] (comentarios) de los usuarios. De esta forma, los prototipos pueden refinarse y el usuario puede obtener una visión más realista del aspecto y la sensación del producto acabado.

Verificación y validación de la implementación real
Cuando los requisitos especifican características de usabilidad para el software (por ejemplo, el número de clics de ratón para alcanzar un objetivo específico), se deben crear casos de prueba para verificar que la implementación del software incluye estas características.

Para realizar la validación de la implementación real, se pueden desarrollar pruebas especificadas para pruebas funcionales de sistema como pruebas de escenarios de usabilidad. Estos escenarios de prueba miden características de usabilidad específicas, como pueden ser la capacidad de ser aprendido o la operabilidad, en vez de los resultados funcionales.

Los escenarios de pruebas para usabilidad pueden desarrollarse para probar de forma específica la sintaxis y las semánticas. La sintaxis es la estructura o la gramática de la interfaz (por ejemplo, qué puede introducirse en un campo de entrada) mientras que la semántica se refiere al significado y el objetivo (por ejemplo, mensajes de sistema razonables y significativos y la salida aportadas al usuario) de la interfaz.

En las pruebas de usabilidad, a veces, se emplean técnicas de caja negra (tales como las descritas en la Sección 3.2), especialmente casos de uso que pueden definirse en texto plano o con UML ("Unified Modeling Language").

Los escenarios de prueba para las pruebas de usabilidad también deben incluir instrucciones para usuarios, asignación de tiempo para entrevistas anteriores y posteriores a la prueba para dar instrucciones y recibir realimentación y un protocolo acordado para realizar las sesiones. Este protocolo incluye una descripción de cómo se llevarán a cabo estas pruebas, tiempos, toma de notas y conexión a la sesión, además de los métodos de entrevista y encuesta que se utilizarán.

Realización de encuestas y cuestionarios
Se pueden utilizar las técnicas de encuesta y cuestionario para recopilar observaciones y realimentación sobre el comportamiento del usuario con el sistema. Algunas encuestas normalizadas y de acceso público como pueden ser IMUS[63] (Inventario de Medición de Usabilidad Software) o "IAMSW[64] (Inventario de Análisis y Medición de Sitios Web)" permiten realizar un análisis comparativo a partir de una base de datos con medidas previas de usabilidad. Además, dado que IMUS aporta medidas de usabilidad concretas, puede ofrecer una conjunto de criterios de aceptación/compleción.

4.2.5 Pruebas de Accesibilidad

Es importante tener en cuenta la accesibilidad que presenta un software para aquellos con necesidades particulares o restricciones para su uso. Esto incluye a aquellos usuarios con discapacidades. Las pruebas de accesibilidad deberían tener en cuenta los estándares relevantes, tales como "Web Content Accesibility Guidelines [65]" o la legislación, como las leyes "Disability Discrimination Acts"[66] (Reino Unido, Australia) y la "Section 508" (Estados Unidos). La accesibilidad, al igual que la usabilidad, debe considerarse durante las fases de diseño. Las pruebas, a menudo, tienen lugar durante los niveles de integración y prosiguen durante las pruebas de sistema y hasta las pruebas de aceptación. En general, los defectos se determinan cuando el software falla en el cumplimiento de las normativas designadas o los estándares definidos para el software.

[62] "realimentación" es la traducción de "feedback".
[63] "IMUS" es la traducción de "SUMI".
[64] "IAMSW" es la traducción de "WAMMI"
[65] Directrices para la Accesibilidad de Contenido Web.
[66] Leyes sobre Discriminación de Discapacidades.

5. Revisiones - 165 minutos

Palabras Clave
ninguna

Objetivos de Aprendizaje para las Revisiones

5.1 Introducción

AP-5.1.1 (K2) Explicar por qué la preparación de las revisiones es importante para el Analista de Pruebas.

5.2 Uso de Listas de Comprobación en las Revisiones

AP-5.2.1 (K4) Analizar un caso de uso o una interfaz de usuario e identificar problemas de acuerdo a la información de listas de comprobación aportada por el programa de estudio.

AP-5.2.2 (K4) Analizar una especificación de requisitos o historia de usuario e identificar problemas de acuerdo a la información de listas de comprobación aportada por el programa de estudio.

5.1 Introducción

Un proceso de revisión correcto requiere planificación, participación y seguimiento. Los Analistas de Pruebas deben participar activamente en el proceso de revisión, aportando su visión particular. Los Analistas de Pruebas deberán contar con preparación formal en revisiones para comprender mejor sus roles respectivos en cualquier proceso de revisión. Todos los participantes de una revisión deben estar comprometidos con los beneficios de una revisión ejecutada de forma correcta. Cuando se hacen correctamente, las revisiones pueden ser la mayor, mejor y más rentable contribución a la calidad general entregada.

Independientemente del tipo de revisión que se realice, el Analista de Pruebas debe permitir el tiempo adecuado para su preparación. Esto incluye tiempo para revisar el producto de trabajo, tiempo para comprobar referencias cruzadas de documentos para verificar la consistencia, y tiempo para establecer que podría faltar del producto de trabajo. Sin un tiempo de preparación adecuado, el Analista de Pruebas podría verse limitado exclusivamente a editar lo que ya está en el documento en lugar de participar en una revisión eficiente que permita maximizar el uso del tiempo del equipo de revisión y ofrecer la mejor realimentación posible. Una buena revisión incluye comprender qué está escrito, determinar qué falta, y verificar que el producto descrito es consistente con otros productos que ya han sido desarrollados o están en desarrollo. Por ejemplo, al revisar un plan de pruebas de nivel de integración, el Analista de Pruebas también debe tener en cuenta los elementos que se están integrando. ¿Qué condiciones deben darse para que estos elementos estén listos para la integración? ¿Hay dependencias que deban ser documentadas? ¿Hay datos disponibles para probar los puntos de integración? Una revisión no se limita al producto de trabajo que se está revisando, sino que también debe tener en cuenta la interacción de dicho elemento con los demás elementos del sistema.

Es fácil que el autor de un producto objeto de una revisión se sienta criticado. El Analista de Pruebas debería asegurarse de abordar cualquier comentario de una revisión desde una perspectiva de colaboración con el autor para crear el mejor producto posible. Con el uso de este enfoque, los comentarios se redactarán de una manera constructiva y estarán orientados hacia el producto de trabajo y no hacia al autor. Por ejemplo, si un enunciado resulta ambiguo, es mejor decir “No entiendo qué es lo que tengo que probar para comprobar que este requisito se ha implementado correctamente. ¿Podría ayudarme a comprenderlo?” en lugar de “Este requisito es ambiguo y nadie será capaz de entenderlo.” El trabajo de un Analista de Pruebas en una revisión es garantizar que la información facilitada en el producto de trabajo será suficiente para soportar el esfuerzo de prueba. Si la información no existe, no es clara o no ofrece el nivel de detalle necesario, entonces es probable que haya un defecto que el autor debe corregir. Al mantener un enfoque positivo en lugar de un enfoque crítico, los comentarios serán mejor recibidos y la reunión será más productiva.

5.2 Uso de Listas de Comprobación en las Revisiones

Las listas de comprobación se utilizan durante las revisiones para recordar a los participantes que tienen que comprobar puntos específicos durante la revisión. Las listas de comprobación también permiten despersonalizar la revisión, por ejemplo, “Esta es la misma lista de comprobación que utilizamos para todas las revisiones, no estamos centrándonos exclusivamente en su producto de trabajo.” Las listas de comprobación pueden ser genéricas y utilizarse para todas las revisiones o pueden concentrarse en características de calidad, áreas o tipos de documentos específicos. Por ejemplo, una lista de comprobación genérica puede comprobar las propiedades del documento general, tales como disponer de un identificador único, no incluir referencias a “pendiente de definir”, tener el formato adecuado e incluir elementos similares de conformidad. Una lista de comprobación específica para un documento de requisitos puede contener comprobaciones sobre el uso correcto de los términos “debe” o “debería”, comprobaciones de la capacidad de ser probado[67] de cada requisito enunciado, etc. El formato de los requisitos también puede indicar el tipo de lista de comprobación que se debe utilizar. Un documento de requisitos que esté en formato de texto narrativo tendrá criterios de revisión distintos de otro que esté basado en diagramas.

Las listas de comprobación también pueden estar orientadas hacia un conjunto de competencias de programador/arquitecto o a un conjunto de competencias de probador. En el caso del Analista de

[67] “capacidad de ser probado” antes “testabilidad”.

Pruebas, la lista de comprobación del conjunto de competencias de probador sería la más adecuada. Estas listas de comprobación pueden incluir elementos tales como los indicados a continuación.

Las listas de comprobación utilizadas para los requisitos, los casos de uso y las historias de usuario normalmente tienen un foco distinto de las empleadas para el código o la arquitectura. Estas listas de comprobación orientadas a los requisitos pueden incluir los siguientes elementos:

- Capacidad de ser probado de cada requisito.
- Criterios de aceptación para cada requisito.
- Disponibilidad de una estructura de llamada del caso de uso, si aplica.
- Identificación única de cada requisito/caso de uso/historia de usuario.
- Versión de cada requisito/caso de uso/historia de usuario.
- Trazabilidad de cada requisito a partir de requisitos de negocio/marketing.
- Trazabilidad entre requisitos y casos de uso.

Lo anterior solo pretende ser un ejemplo. Es importante recordar que si un requisito no presenta la capacidad de ser probado, es decir, si está definido de tal forma que el Analista de Pruebas no puede determinar cómo probarlo, entonces hay un defecto en dicho requisito. Por ejemplo, un requisito que dice “El software debería ser muy amigable para el usuario” no tiene capacidad de ser probado. ¿Cómo puede el Analista de Pruebas determinar si el software es amigable, o incluso muy amigable para el usuario? Si, al contrario, el mismo requisito dijera “El software debe ajustarse a los estándares de usabilidad previstos en el documento de estándares de usabilidad”, y si el documento de estándares de usabilidad existiera de verdad, entonces sí sería un requisito con capacidad de ser probado. También se trata de un requisito general porque este mismo requisito se aplica a todos los elementos de la interfaz. En este caso, este mismo requisito podría fácilmente dar lugar a casos de prueba individuales en una aplicación no trivial. La trazabilidad a partir de este requisito, o quizás a partir del documento de estándares de usabilidad hasta los casos de prueba también es crítica ya que, si la especificación de usabilidad referenciada cambiara, se tendrían que revisar y actualizar todos los casos de prueba en consecuencia.

Se considera que un requisito no presenta capacidad de ser probado si el probador es incapaz de establecer si este ha pasado o no la prueba, o no puede construir una prueba que este pueda pasar o fallar. Por ejemplo, “El sistema debe estar disponible el 100% del tiempo, 24 horas al día, 7 días a la semana, 365 (o 366) días al año” no tiene capacidad de ser probado.

.

Una sencilla lista de comprobación para revisiones de casos de uso puede incluir las siguientes preguntas:

- ¿El camino (escenario) principal está definido de forma clara?
- ¿Están identificados todos los caminos (escenarios) alternativos, completos con tratamiento de errores?
- ¿Están definidos los mensajes de la interfaz de usuario?
- ¿Hay un único camino principal (de lo contrario debería haber varios casos de uso)?
- ¿Todos los caminos tienen capacidad de ser probados?

Una sencilla lista de comprobación de usabilidad para la interfaz de usuario de una aplicación puede incluir:

- ¿Están definidos todos los campos y su función?
- ¿Están definidos todos los mensajes de error?
- ¿Están definidos todos los avisos[68] de usuario y son consistentes?
- ¿Está definido el orden de tabulación de los campos?
- ¿Hay alternativas de teclado a las acciones del ratón?
- ¿Hay métodos abreviados[69] de combinaciones de teclas definidas para el usuario (por ejemplo, cortar y pegar)?
- ¿Hay dependencias entre los campos (tales como una determinada fecha tiene que ser posterior a otra fecha)?
- ¿Hay un diseño de pantalla?

[68] “aviso de usuario” es la traducción de “user prompt”.
[69] “método abreviado” es la traducción de “shortcut”.

- ¿El diseño de pantalla concuerdan con los requisitos especificados?
- ¿Hay algún indicador para el usuario que aparezca cuando el sistema está procesando?
- ¿La pantalla cumple con el requisito del número de clics de ratón mínimo[70] (si está definido)?
- ¿La navegación fluye de una forma lógica para el usuario en base a la información del caso de uso?
- ¿La pantalla cumple con los requisitos relacionados con la capacidad de ser aprendido?
- ¿Hay texto de ayuda disponible para el usuario?
- ¿Hay texto flotante[71] disponible para el usuario?
- ¿El usuario considerará que esto es "atractivo" (evaluación subjetiva)?
- ¿El uso de colores es consistente con otras aplicaciones y con los estándares de la organización?
- ¿Se están utilizando correctamente los efectos de sonido y son configurables?
- ¿La pantalla cumple con los requisitos de regionalización[72]?
- ¿El usuario puede decidir qué hacer (capacidad de ser entendido) (evaluación subjetiva)?
- ¿El usuario podrá recordar qué hacer (capacidad de ser aprendido) (evaluación subjetiva)?

En un proyecto Ágil, los requisitos normalmente adoptan la forma de historias de usuario. Estas historias representan pequeñas unidades de funcionalidad demostrable. Mientras que un caso de uso es una transacción de usuario que atraviesa varias áreas de funcionalidad, una historia de usuario está más aislada y su alcance generalmente se establece en función del tiempo necesario para su desarrollo. Una lista de comprobación para una historia de usuario puede incluir las siguientes preguntas:

- ¿La historia es adecuada para la iteración/sprint objetivo?
- ¿Los criterios de aceptación están definidos y son susceptibles de ser probados?
- ¿La funcionalidad está claramente definida?
- ¿Existe alguna dependencia entre esta y otras historias?
- ¿La historia está priorizada?
- ¿La historia contiene un elemento de funcionalidad?

Por supuesto, si la historia define una nueva interfaz, entonces debería utilizarse una lista de comprobación genérica de historias (tal como la anterior) y una lista de comprobación de interfaz de usuario detallada.

Las listas de comprobación se pueden personalizar basándose en lo siguiente:

- Organización (por ejemplo, tener en cuenta las políticas, los estándares y los convenciones de la organización).
- Proyecto/ esfuerzo de desarrollo (por ejemplo, centro de atención, estándares técnicos, riesgos).
- Objeto sujeto a revisión (por ejemplo, las revisiones de código pueden adaptarse a lenguajes de programación específicos).

Las listas de comprobación buenas detectarán problemas y ayudarán a abrir debates sobre otros elementos que puede que no estén específicamente referenciados en la lista de comprobación. El uso de una combinación de listas de comprobación es una forma eficiente de garantizar que una revisión logra un producto de trabajo de la máxima calidad. El uso de listas de comprobación estándar, tales como las previstas en el programa de estudio "Probador Certificado – Nivel Básico", y de listas de comprobación desarrolladas específicamente para las organizaciones, tales como las indicadas más arriba, ayudarán al Analista de Pruebas a ser efectivo en las revisiones.

Para más información sobre revisiones e inspección, consulte [Gilb93] y [Weigers03].

[70] "requisito del número de clics de ratón mínimo" es la traducción de "minimum mouse click requirement". Requisito en el que se define el número de clics de ratón aceptable.
[71] "texto flotante" es la traducción de "hover text".
[72] "regionalización" es la traducción de "localization". También se utiliza el término "localización".

6. Gestión de Defectos – [120 minutos]

Palabras Clave
taxonomía de defectos, contención de fase, análisis de la causa raíz

Objetivos de Aprendizaje para Gestión de Defectos

6.2 ¿Cuándo se Puede Detectar un Defecto?
AP-6.2.1 (K2) Explicar cómo la contención de fase puede reducir los costes.

6.3 Campos del Informe de Defecto
AP-6.3.1 (K2) Explicar la información que puede ser necesaria para documentar un defecto no funcional.

6.4 Clasificación del Defecto
AP-6.4.1 (K4) Identificar, recopilar y registrar la información de clasificación de un defecto dado.

6.5 Análisis de la Causa Raíz
AP-6.5.1 (K2) Explicar el objeto del análisis de la causa raíz.

6.1 Introducción

Los Analistas de Pruebas evalúan el comportamiento en términos de las necesidades de negocio y de los usuarios, por ejemplo, si un usuario sabría lo que tiene que hacer si se encontrara con determinado mensaje o comportamiento. Al comparar el resultado real con el resultado esperado, el Analista de Pruebas determina si el sistema se está comportando correctamente. Una anomalía (también denominada incidencia) es un hecho inesperado que requiere investigación adicional. Una anomalía puede ser un fallo provocado por un defecto. Una anomalía puede o no resultar en la redacción de un informe de defecto. Un defecto es un problema real que se debería resolver.

6.2 ¿Cuándo se Puede Detectar un Defecto?

Los defectos pueden detectarse a través de pruebas estáticas y los síntomas de los defectos, los fallos, pueden detectarse mediante pruebas dinámicas. Cada fase del ciclo de vida de desarrollo software debe disponer de métodos para detectar y eliminar posibles fallos. Por ejemplo, durante la fase de desarrollo, se deberían utilizar revisiones de código y diseño para detectar defectos. Durante las pruebas dinámicas, se utilizan casos de prueba para la detección de fallos.

Cuanto antes se detecte y corrija un defecto, más bajo resultará el coste de la calidad para el sistema en general. Por ejemplo, las pruebas estáticas pueden encontrar defectos antes de que se puedan realizar pruebas dinámicas. Esta es uno de las razones por los que las pruebas estáticas constituyen un enfoque rentable para producir software de alta calidad.

El sistema de seguimiento de defectos debería permitir al Analista de Pruebas registrar la fase del ciclo de vida en la que se introdujo el defecto y la fase en la que se detectó. Si las dos fases coinciden, entonces se ha logrado una contención de fase perfecta. Esto significa que el defecto se introdujo y se detectó en la misma fase, y no se “escapó” a una fase posterior. Un ejemplo de esto sería la identificación de un requisito incorrecto durante la revisión de requisitos y su corrección en ese momento. Este hecho no solo constituye un uso eficiente de la revisión de requisitos, sino que también evita que el defecto dé lugar a trabajo adicional que encarecería el coste para la organización. Si un requisito incorrecto se “escapa” de la revisión de requisitos y, en consecuencia, el desarrollador lo implementa, el Analista de Pruebas lo prueba, y el usuario lo detecta durante las pruebas de aceptación del usuario, todo el trabajo que se haya realizado sobre dicho requisito habrá sido una pérdida de tiempo (sin mencionar que es posible que el usuario pierda la confianza en el sistema).

La contención de fase es una forma efectiva de reducir el coste de los defectos.

6.3 Campos del Informe de Defecto

Los campos (parámetros) facilitados para el informe de defecto tienen por objeto facilitar información suficiente para que el informe de defecto sea susceptible de procesamiento[73]. Un informe de defectos susceptible de procesamiento es:

- Completo - toda la información necesaria está en el informe.
- Conciso - el informe no incluye información superflua[74].
- Exacto - la información del informe es correcta e indica claramente los resultados reales y esperados así como los pasos adecuados para reproducir.
- Objetivo - el informe es una exposición de los hechos[75] redactada de forma profesional.

La información registrada en un informe de defecto debe dividirse en campos de datos. Cuanto mejor estén definidos los campos, más fácil será informar los defectos individuales y elaborar informes de tendencia y otros informes resumen. Cuando se dispone de un número definido de opciones para un campo, contar con listas desplegables de los valores disponibles puede reducir el tiempo necesario para registrar un defecto. Las listas desplegables[76] sólo son efectivas cuando el número de opciones

[73] “susceptible de procesamiento” es la traducción de “actionable”.
[74] “superflua” es la traducción de “extraneous”.
[75] “exposición de los hechos” es la traducción de “statement of fact”.
[76] “lista desplegable” es la traducción de “drop down list”.

es limitado y el usuario no tiene que desplazarse[77] a lo largo de una lista extensa para encontrar la opción correcta. Los distintos tipos de informes de defecto requieren información diferente, por lo que la herramienta de gestión de defectos debería ser lo suficientemente flexible como para mostrar los campos correctos en función del tipo de defecto. Los datos deben registrarse en campos distintos, idealmente soportados por un proceso de validación de datos para evitar entradas de datos fallidas y garantizar la gestión efectiva de la información (informes).

Los informes de defecto se elaboran para fallos descubiertos durante las pruebas funcionales y no funcionales. La información incluida en un informe de defecto debería estar orientada en todo momento a identificar claramente el escenario en el que se ha detectado el problema, incluyendo los pasos y los datos necesarios para reproducir dicho escenario, así como los resultados esperados y reales. Los informes de defecto no funcionales pueden requerir más detalles sobre el entorno, otros parámetros de rendimiento (por ejemplo, las tamaño de la carga), la secuencia de los pasos y los resultados esperados. Al documentar un fallo de usabilidad, es importante establecer qué esperaba el usuario que hiciera el software. Por ejemplo, si el estándar de usabilidad es que una operación debe completarse en menos de cuatro clics de ratón, el informe de defectos debería indicar cuántos clics fueron necesarios frente al estándar previsto. En los casos en los que no se disponga de un estándar y los requisitos no cubran los aspectos de calidad no funcional del software, el probador podrá recurrir a una prueba de la “persona razonable”[78] para establecer si la usabilidad es aceptable o no. En ese caso, las expectativas de esa “persona razonable” se deben indicar claramente en el informe de defecto. Dado que a veces faltan requisitos no funcionales en la documentación de requisitos, la documentación de fallos no funcionales plantea más retos al probador al documentar el comportamiento “esperado” frente al comportamiento “real”.

Si bien el objetivo habitual de la redacción de un informe de defecto es obtener una solución al problema, la información del defecto también es necesaria para una clasificación, análisis de riesgos y mejora de los procesos exactos.

6.4 Clasificación del Defecto

Existen varios niveles de clasificación de los que un informe de defecto puede ser objeto a lo largo de su ciclo de vida. La correcta clasificación de un defecto es una parte integral de la gestión de la información del defecto. Las clasificaciones sirven para agrupar los defectos, evaluar la efectividad de las pruebas, evaluar la efectividad del ciclo de vida de desarrollo y establecer tendencias interesantes.

Entre la información de clasificación típica para defectos recién identificados se encuentra:

- Actividad del proyecto que tuvo como resultado la detección del defecto - por ejemplo, revisión, auditoría, inspección, codificación, pruebas.
- Fase del proyecto en la que se introdujo el defecto (si se conoce) - por ejemplo, requisitos, diseño, diseño detallado, código.
- Fase del proyecto en la que se ha detectado el defecto - por ejemplo, diseño, diseño detallado, código, revisión de código, pruebas unitarias, pruebas de integración, pruebas de sistema, pruebas de aceptación.
- Presunta causa del defecto - por ejemplo, requisitos, diseño, interfaz, código, datos.
- Repetibilidad - por ejemplo, una vez, intermitente, reproducible.
- Síntoma - por ejemplo, fallo abrupto[79], bloqueo[80], error de interfaz de usuario, error de sistema, rendimiento.

Una vez investigado el defecto, puede ser posible una clasificación adicional:

- Causa raíz - el error cometido que tuvo como resultado el defecto, por ejemplo, proceso, error de codificación, error de usuario, error de prueba, problema de configuración, problema de las datos, software de un tercero, problema de software externo, problema de documentación.

[77] “desplazarse” es la traducción de “scroll”.
[78] “persona razonable” es la traducción de “reasonable person”.
[79] “fallo abrupto” es la traducción de “crash”.
[80] “bloqueo” es la traducción de “hang”.

- Fuente - el producto de trabajo en el que se ha producido el error, por ejemplo, requisitos, diseño, diseño detallado, arquitectura, diseño de base de datos, documentación de usuario, documentación de la prueba.
- Tipo - por ejemplo, problema lógico, problema de cálculo, problema de tiempo, tratamiento de datos, mejora.

Cuando el defecto es corregido (o se ha diferido o ha fallado la confirmación), puede haber, incluso, aún más información de clasificación disponible, como:

- Resolución - por ejemplo, cambio en código, cambio de documentación, diferido, no es un problema, duplicado.
- Acción correctiva - por ejemplo, revisión de requisitos, revisión de código, prueba unitaria, documentación de configuración, preparación de datos, no ha habido cambio.

Además de estas categorías de clasificación, a menudo, los defectos también se clasifican en base a su severidad y prioridad. Además, en función del proyecto, puede tener sentido clasificar en base al impacto en la seguridad física de la misión, el impacto en el calendario del proyecto, los costes del proyecto, el riesgo del proyecto e impacto en la calidad del proyecto. Estas clasificaciones pueden tenerse en cuenta en contratos por lo que respecta a la rapidez con la que se entregará la corrección.

La última área de clasificación es la resolución definitiva. A menudo los defectos se agrupan en base a su resolución, por ejemplo, corregido/verificado, cerrado/no es un problema, diferido, abierto/no resuelto. Normalmente esta clasificación se utiliza durante todo el proyecto, ya que se hace un seguimiento de los defectos a lo largo de todo su ciclo de vida.

A menudo los valores de clasificación empleados por una organización están personalizados. Lo indicado con anterioridad sólo corresponde a ejemplos de algunos de los valores comunes empleados en el sector. Es importante que los valores de clasificación se utilicen de manera consistente para que sean útiles. Demasiados campos de clasificación harán que la apertura y el procesamiento de un defecto requieran mucho tiempo, por lo que es importante ponderar el valor de los datos recopilados frente al coste incremental de cada defecto procesado. La capacidad para personalizar los valores de clasificación recopilados por una herramienta, a menudo, constituye un factor importante en la selección de herramientas.

6.5 Análisis de la Causa Raíz

El objetivo del análisis de la causa raíz es determinar qué provocó la ocurrencia del defecto y aportar datos que respalden los cambios necesarios en el proceso que eliminarán las causas raíz que son responsables de una parte importante de los defectos. El análisis de la causa raíz, normalmente, lo realiza la persona que investiga o bien soluciona el problema o determina que el problema no debería o no puede solucionarse. Suele tratarse del desarrollador. Normalmente el Analista de Pruebas establece un valor preliminar para la causa raíz con el fin de hacer una suposición fundamentada[81] sobre qué fue lo que probablemente provocó el problema. Cuando tenga lugar la confirmación de la corrección, el Analista de Pruebas verificará el valor de la causa raíz introducido por el desarrollador. En el momento en que se determina la causa raíz, también es habitual determinar o confirmar la fase en la que se introdujo el defecto.

Algunas causas raíz típicas son:

- Requisito poco claro.
- Requisito ausente.
- Requisito erróneo.
- Implementación incorrecta del diseño.
- Implementación incorrecta de la interfaz.
- Error lógico de código.
- Error de cálculo.
- Error de hardware.
- Error de interfaz.

[81] “suposición fundamentada” es la traducción de “educated guess”.

- Datos no válidos.

Esta información sobre la causa raíz se agrega para establecer problemas comunes que resultan en la creación de defectos. Por ejemplo, si hay un gran número de defectos provocados por requisitos poco claros, tendría sentido aplicar más esfuerzo en realizar revisiones de requisitos efectivas. Del mismo modo, si la implementación de interfaz supone un problema en los grupos de desarrollo, es posible que sea necesario celebrar sesiones conjuntas de diseño.

El uso de la información de causas raíz para la mejora de procesos ayuda a la organización a realizar un seguimiento de las ventajas de los cambios de proceso efectivos y cuantificar los costes de los defectos que pueden atribuirse a una causa raíz en particular. Esto puede ayudar a obtener financiación para los cambios en procesos que puedan requerir la compra de herramientas y equipos adicionales, así como la modificación del calendario de plazos establecido. El análisis de la causa raíz se estudia en mayor detalle en el programa de estudio de Nivel Experto del ISTQB “Mejora del proceso de prueba” [ISTQB_EL_ITP].

7. Herramientas de Prueba - 45 minutos

Palabras Clave

pruebas guiadas por palabra clave, herramienta de preparación de datos de prueba, herramienta de diseño de pruebas, herramienta de ejecución de pruebas

Objetivos de Aprendizaje de Herramientas de Prueba

7.2 Herramientas de Prueba y Automatización

AP-7.2.1 (K2) Explicar las ventajas de utilizar herramientas de preparación de datos de prueba, herramientas de diseño de pruebas y herramientas de ejecución de pruebas.

AP-7.2.2 (K2) Explicar el papel del Analista de Pruebas en la automatización guiada por palabra clave.

AP-7.2.3 (K2) Explicar los pasos para solucionar un fallo de ejecución de prueba automatizada.

7.1 Introducción

Las herramientas de prueba pueden mejorar significativamente la eficiencia y exactitud del esfuerzo de prueba, pero sólo si implementan las herramientas adecuadas en la forma adecuada. Las herramientas de prueba deben ser tratadas como otro aspecto a tener en cuenta en una organización de prueba bien gestionada. La sofisticación y aplicabilidad de las herramientas de prueba son muy variadas y el mercado de herramientas está en constante evolución. Normalmente las herramientas se adquieren de proveedores de herramientas comerciales y de distintos sitios de herramientas software gratuito[82] o software compartido[83].

7.2 Herramientas de Prueba y Automatización

Gran parte del trabajo de un Analista de Pruebas requiere el uso efectivo de herramientas. Saber qué herramientas utilizar, y cuándo utilizarlas, puede aumentar la eficiencia del Analista de Pruebas y puede ayudar a ofrecer una mejor cobertura de pruebas dentro del tiempo permitido.

7.2.1 Herramientas de Diseño de Pruebas

Las herramientas de diseño de pruebas sirven para ayudar a crear casos de prueba y datos de prueba que se utilizarán en pruebas. Estas herramientas pueden funcionar a partir de formatos de documentos de requisitos específicos, modelos (por ejemplo, UML) o entradas facilitadas por el Analista de Pruebas. Las herramientas de diseño de pruebas, a menudo, se diseñan y construyen para funcionar con formatos específicos y con productos específicos, tales como herramientas de gestión de requisitos específicos.

Las herramientas de diseño de pruebas pueden facilitar información al Analista de Pruebas para determinar qué tipos de pruebas deben realizarse para obtener el nivel objetivo de cobertura de prueba, confianza en el sistema o medidas para mitigar de riesgos de producto. Por ejemplo, las herramientas de árbol de clasificación generan (y muestran) la serie de combinaciones que se necesita para alcanzar una cobertura completa tomando como base un criterio de cobertura seleccionado. El Analista de Pruebas podrá emplear esta información más adelante para determinar los casos de prueba que deben ejecutarse.

7.2.2 Herramientas de Preparación de Datos de Prueba

Las herramientas de preparación de datos de prueba aportan varias ventajas. Algunas herramientas de preparación de datos de prueba pueden analizar un documento, como por ejemplo un documento de requisitos o incluso el código fuente, para determinar los datos que se necesitan durante las pruebas para alcanzar un nivel de cobertura. Otras herramientas de preparación de datos de prueba pueden recoger un conjunto de datos de un sistema de producción y “limpiarlos”[84] o “disociarlos”[85] para eliminar cualquier información personal sin perder la integridad interna de los datos. A continuación, los datos disociados pueden utilizarse para realizar pruebas sin correr el riesgo de una fuga de seguridad o de un uso incorrecto de información personal. Esto es de particular importancia cuando se requieren grandes volúmenes de datos reales. Otras herramientas de generación de datos permiten generar datos de prueba a partir de conjuntos dados de parámetros de entrada (es decir, para su uso en pruebas aleatorias). Algunas de estas herramientas analizarán la estructura de la base de datos para determinar qué entradas deberá aportar el Analista de Pruebas.

7.2.3 Herramientas de Ejecución de Pruebas Automatizadas

Las herramientas de ejecución de pruebas son utilizadas, fundamentalmente, por los Analistas de Pruebas en todos los niveles de prueba para ejecutar pruebas y comprobar los resultados de estas pruebas. El objetivo de emplear una herramienta de ejecución de pruebas es normalmente uno o varios de los siguientes:

[82] “software gratuito” es la traducción de “freeware”.
[83] “software compartido” es la traducción de “shareware”.
[84] “limpiar” es la traducción de “scrub”.
[85] “disociar” es la traducción de “anonymize”.

- Reducir los costes (en términos de esfuerzo y/o tiempo).
- Ejecutar más pruebas.
- Ejecutar la misma prueba en muchos entornos.
- Hacer que la ejecución de las pruebas se más repetible.
- Ejecutar pruebas que sería imposible llevar a cabo manualmente (es decir, pruebas para la validación de grandes cantidades de datos).

Estos objetivos a menudo se solapan con los objetivos principales de aumentar la cobertura reduciendo, a la vez, los costes.

7.2.3.1 Aplicabilidad

El retorno sobre la inversión para las herramientas de ejecución de pruebas, normalmente, es más alto cuando se automatizan las pruebas de regresión, dado el bajo nivel de mantenimiento esperado y la ejecución repetida de las pruebas. La automatización de las pruebas de humo también puede ser un uso efectivo de la automatización debido al uso frecuente de las pruebas, la necesidad de un resultado rápido y, a pesar de que el coste de mantenimiento puede ser más alto, la capacidad de disponer de una forma automatizada de evaluar una nueva versión en un entorno de integración continua.

Las herramientas de ejecución de pruebas, con frecuencia, se utilizan durante los niveles de pruebas de sistema e integración. Algunas herramientas, en particular las herramientas de pruebas de API, también se pueden utilizar en el nivel de pruebas de componente. Fomentar el uso de estas herramientas donde sea más apropiado ayudará a mejorar el retorno sobre la inversión.

7.2.3.2 Fundamentos de Herramientas de Automatización de Pruebas

Las herramientas de ejecución de pruebas funcionan ejecutando un conjunto de instrucciones escritas en un lenguaje de programación, también conocido como lenguaje de guion[86]. Las instrucciones para la herramienta se encuentran a un nivel muy detallado que especifica entradas, orden de la entrada, valores específicos utilizados para las entradas y las salidas esperadas. Esto puede hacer que los guiones detallados sean susceptibles a los cambios en el software sujeto a prueba (SSP), especialmente cuando la herramienta está interactuando con la interfaz gráfica de usuario (IGU[87]).

La mayoría de las herramientas de ejecución incorporan un comparador, que ofrece la capacidad de comparar un resultado real con un resultado esperado almacenado.

7.2.3.3 Implementación de la Automatización de Pruebas

La tendencia en la automatización de la ejecución de las pruebas (como en la programación) es pasar de instrucciones detalladas a bajo nivel a lenguajes de alto nivel, empleando librerías, macros y subprogramas. Las técnicas de diseño tales como las pruebas guiadas por palabra clave y las pruebas guiadas por palabra de acción capturan una serie de instrucciones y las referencian con una “palabra clave” o “palabra de acción” particulares. Esto permite al Analista de Pruebas redactar los casos de prueba en un lenguaje humano mientras se ignora el lenguaje de programación subyacente y de las funciones de nivel inferior. El uso de esta técnica de escritura modular permite una mantenibilidad más sencilla durante los cambios en la funcionalidad y la interfaz del software sujeto a prueba. [Bath08] El uso de palabras clave en guiones automatizados se aborda a continuación.

Se pueden utilizar modelos para guiar la creación de las palabras clave o las palabras de acción. Observando los modelos de proceso de negocio que, a menudo, se incluyen en los documentos de requisitos, el Analista de Pruebas puede determinar los procesos de negocio clave que se deben probar. Entonces se pueden determinar los pasos para estos procesos, incluyendo los puntos de decisión que pueden producirse durante los procesos. Los puntos de decisión pueden convertirse en palabras de acción que la automatización de las pruebas puede obtener y utilizar a partir de las hojas de cálculo de palabras clave o palabras de acción. El modelado de procesos de negocio es un método de documentación de los procesos de negocio y que se puede utilizar para identificar estos procesos y

[86] “lenguaje de guion” es la traducción de “scripting language”.
[87] “IGU” es el acrónimo de “Interfaz Gráfica de Usuario”. “IGU” es el equivalente a “GUI” en inglés.

puntos de decisión clave. El modelado puede realizarse manualmente o mediante herramientas que actúan sobre las entradas tomando como base las reglas de negocio y descripciones de procesos.

7.2.3.4 Mejora del Éxito del Esfuerzo de Automatización

Al determinar qué pruebas automatizar, se deben evaluar cada uno de los casos de prueba candidatos o juegos de pruebas candidatos para ver si merecen ser automatizados. Muchos proyectos de automatización, poco exitosos, están basados en la automatización de casos de prueba manuales más fáciles de obtener, sin comprobar el beneficio real de la automatización. Puede ser que, para un conjunto de casos de prueba (un juego) dado, resulte óptimo incluir pruebas manuales, semiautomatizadas y automatizadas.

Cuando se implemente un proyecto de automatización de ejecución de pruebas, se deben tener en cuenta los siguientes aspectos:

Posibles ventajas

- El tiempo de ejecución de las pruebas automatizadas pasará a ser más previsible.
- Las pruebas de regresión y la validación de defectos mediante pruebas automatizadas serán más rápidas y más fiables en etapas tardías del proyecto.
- El estatus y desarrollo técnico del probador o del equipo de pruebas puede mejorar con el uso de herramientas automatizadas.
- La automatización puede ser especialmente útil en ciclos de vida de desarrollo iterativos e incrementales para ofrecer mejores pruebas de regresión para cada versión o iteración.
- Es posible que la cobertura de determinados tipos de prueba sólo se pueda alcanzar con herramientas automatizadas (por ejemplo, esfuerzos de validación para grandes cantidades de datos).
- La automatización de la ejecución de pruebas puede ser más rentable que las pruebas manuales para grandes cantidades de datos de entrada, la conversión y comparación de los esfuerzos de prueba aportando entradas y verificaciones rápidas y consistentes.

Posibles riegos:

- Las pruebas manuales incompletas, inefectivas o incorrectas se pueden automatizar “en el estado actual”[88].
- El mantenimiento de los productos de prueba[89] puede resultar difícil, requiriendo múltiples modificaciones cuando el software sujeto a prueba cambia.
- La participación directa del probador en la ejecución de las pruebas puede verse reducida, dando lugar a una menor detección de defectos.
- El equipo de pruebas puede no tener aptitudes suficientes para emplear las herramientas de automatización de una forma efectiva.
- Las pruebas irrelevantes que no contribuyen a la cobertura general de la prueba pueden ser automatizadas porque existen y son estables.
- Las pruebas pueden pasar a ser improductivas a medida que el software se estabiliza (paradoja del pesticida).

Durante el despliegue de una herramienta de automatización de prueba, no siempre es acertado automatizar casos de prueba en su estado actual, sino redefinir los casos de prueba para hacer un mejor uso de la automatización. Ello incluye formatear los casos de prueba, considerar la posibilidad de reutilizar patrones, ampliar las entradas mediante el uso de variables en lugar de usar valores incrustados[90] y hacer uso de todas las ventajas que ofrece la herramienta de pruebas. Las herramientas de pruebas suelen tener la capacidad de atravesar múltiples pruebas, agrupar pruebas, repetir pruebas y cambiar el orden de ejecución, mientras aporta facilidades de análisis y gestión de la información.

Para muchas de las herramientas de automatización de ejecución de pruebas, se requieren determinadas habilidades para la programación, para crear pruebas (guiones) y juegos de prueba

[88] “en el estado actual” es la traducción de “as is”.
[89] “productos de prueba” antes “producto de soporte de prueba”.
[90] “incrustado” es la traducción de “hard-coded”.

eficientes y efectivos. Es muy habitual que juegos de prueba automatizados, de gran tamaño, sean difíciles de actualizar y gestionar si no se diseñan con cuidado. Una formación adecuada en herramientas de prueba, programación y técnicas de diseño es valiosa para asegurar que se aprovecha el máximo de los beneficios de las herramientas.

Durante la planificación de la prueba, es importante dejar tiempo para ejecutar periódicamente los casos de prueba automatizados de forma manual con el fin de retener el conocimiento de cómo funciona la prueba y comprobar una operación correcta, así como para revisar la validez de los datos de entrada y la cobertura.

7.2.3.5 Automatización Guiada por Palabra Clave

Las palabras claves (a veces llamadas “palabras de acción”), se utilizan principalmente (aunque no exclusivamente) para representar las interacciones de alto nivel de negocio con un sistema (por ejemplo, “cancelar orden”). Cada palabra clave se utiliza para representar una serie de interacciones detalladas entre un actor y el sistema sujeto a prueba. Para la especificación de casos de prueba se emplean secuencias de palabras clave (incluyendo datos de pruebas relevantes). [Buwalda01]

En la automatización de pruebas, una palabra clave se implementa como uno o más guiones de prueba ejecutables. Las herramientas leen los casos de prueba redactados como una secuencia de palabras clave que invocan a los guiones de prueba adecuados que implementan la funcionalidad de la palabra clave. Los guiones se implementan de una forma altamente modular para permitir establecer fácilmente la correspondencia con palabras claves específicas. Se requieren habilidades de programación para poder implementar estos guiones modulares.

Las ventajas fundamentales de las pruebas guiadas por palabra clave son las siguientes:

- Las palabras clave que se refieren a una aplicación o a un dominio de negocio particular pueden definirlas expertos de dominio. Esto puede hacer que la tarea de especificación de caso de prueba se realice de forma más eficiente.
- Una persona que, fundamentalmente, cuente con un conocimiento experto del dominio se podrá beneficiar de la ejecución de los casos de prueba automáticos (una vez que las palabras claves hayan sido implementadas como guiones) sin tener que entender el código de automatización subyacente.
- El mantenimiento de los casos de prueba redactados con palabras clave es más sencillo, puesto que es menos probable que necesiten modificaciones si cambian los detalles del software sujeto a prueba.
- Las especificaciones de los casos de prueba son independientes de su implementación. Las palabras clave se pueden implementar empleando una gran variedad de lenguajes de creación de guiones y herramientas.

Los desarrolladores o los Analistas de Pruebas Técnicas son los que, normalmente, redactan los guiones de automatización (el código de automatización efectivo) que utilizan información de palabras clave/palabras de acción, mientras que los Analistas de Pruebas son los que, normalmente, crean o mantienen los datos de palabras clave/palabras de acción. Mientras que la automatización guiada por palabras clave normalmente se ejecuta durante la fase de pruebas de sistema, el desarrollo del código puede iniciarse en las primeras etapas de integración. En un entorno iterativo, el desarrollo de la automatización de las pruebas es un proceso continuo.

Una vez creadas las palabras clave de entrada y los datos, los Analistas de Pruebas generalmente asumen la responsabilidad de ejecutar los casos de prueba guiados por palabras clave y analizar cualquier fallo que pueda producirse. Cuando se detecta una anomalía, el Analista de Pruebas debe investigar la causa del fallo para determinar si el problema está en las palabras clave, en los datos de entrada, en el propio guion de automatización o en la aplicación objeto de prueba. Normalmente el primer paso para la resolución de problemas es ejecutar la misma prueba con los mismos datos, de forma manual, para ver si el fallo se encuentra en la propia aplicación. Si no se detecta ningún fallo, el Analista de Pruebas debe revisar la secuencia de pruebas que ha dado lugar al fallo para determinar si el problema se ha producido en un paso anterior (tal vez, al producir datos incorrectos), pero el problema no se deja ver hasta más avanzado el proceso. Si el Analista de Pruebas no puede determinar la causa

del fallo, la información de la resolución del problema debería devolverse al Analista de Pruebas Técnicas o al desarrollador para su posterior análisis.

7.2.3.6 Causas de Fallos en el Esfuerzo de Automatización

Los proyectos de automatización de la ejecución de pruebas, con frecuencia, fallan en la consecución de sus objetivos. Estos fallos pueden deberse a una flexibilidad insuficiente en el uso de la herramienta de pruebas, a la falta de competencias de programación del equipo de pruebas o a unas expectativas poco realistas de los problemas que se pueden resolver mediante la automatización de la ejecución de las pruebas. Es importante tener en cuenta que cualquier automatización de la ejecución de pruebas supone gestión, esfuerzo, pericia y atención, al igual que cualquier proyecto de desarrollo de software. Es necesario dedicar tiempo a crear una arquitectura sostenible, siguiendo prácticas de diseño adecuadas, aportando una gestión de la configuración y aplicando buenas prácticas de codificación. Los guiones de pruebas automatizadas tienen que probarse porque es probable que contengan defectos. Es posible que los guiones tengan que ajustarse a efectos del rendimiento. La usabilidad de las herramientas también debe tenerse en cuenta, no solo en lo que respecta al desarrollador sino también en lo que respecta a las personas que utilizarán la herramienta para ejecutar guiones. Es posible que se deba diseñar una interfaz, entre la herramienta y el usuario, que aportará acceso a los casos de prueba de tal manera que estén organizados de forma lógica para el probador sin dejar de ofrecer la accesibilidad que requiere la herramienta.

8. Referencias

8.1 Estándares

- [ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)
  Capítulos 1 y 4
- [ISO9126] ISO/IEC 9126-1:2001, Software Engineering - Software Product Quality,
  Capítulos 1 y 4
- [RTCA DO-178B/ED-12B]: Software Considerations in Airborne Systems and Equipment Certification, RTCA/EUROCAE ED12B.1992.
  Capítulo 1

8.2 Documentos ISTQB

- [ISTQB_AL_OVIEW] ISTQB Advanced Level Overview, Version 1.0
- [ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 1.0
- [ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 1.0
- [ISTQB_FL_SYL] ISTQB Foundation Level Syllabus, Version 2011
- [ISTQB_GLOSSARY] Standard glossary of terms used in Software Testing, Version 2.2, 2012

8.3 Referencias Bibliográficas

[Bath08] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook”, Rocky Nook, 2008, ISBN 978-1-933952-24-6

[Beizer95] Boris Beizer, “Black-box Testing”, John Wiley & Sons, 1995, ISBN 0-471-12094-4

[Black02]: Rex Black, “Managing the Testing Process (2nd edition)”, John Wiley & Sons: New York, 2002, ISBN 0-471-22398-0

[Black07]: Rex Black, “Pragmatic Software Testing”, John Wiley and Sons, 2007, ISBN 978-0-470-12790-2

[Buwalda01]: Hans Buwalda, “Integrated Test Design and Automation”, Addison-Wesley Longman, 2001, ISBN 0-201-73725-6

[Cohn04]: Mike Cohn, “User Stories Applied: For Agile Software Development”, Addison-Wesley Professional, 2004, ISBN 0-321-20568-5

[Copeland03]: Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2003, ISBN 1-58053-791-X

[Craig02]: Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9

[Gerrard02]: Paul Gerrard, Neil Thompson, “Risk-based e-business Testing”, Artech House, 2002, ISBN 1-580-53314-0

[Gilb93]: Tom Gilb, Graham Dorothy, “Software Inspection”, Addison-Wesley, 1993, ISBN 0-201-63181-4

[Graham07]: Dorothy Graham, Erik van Veenendaal, Isabel Evans, Rex Black “Foundations of Software Testing”, Thomson Learning, 2007, ISBN 978-1-84480-355-2

[Grochmann94]: M. Grochmann (1994), Test case design using Classification Trees, in: conference proceedings STAR 1994

[Koomen06]: Tim Koomen, Leo van der Aalst, Bart Broekman, Michiel Vroon "TMap NEXT, for result driven testing", UTN Publishers, 2006, ISBN 90-72194-80-2

[Myers79]: Glenford J. Myers, “The Art of Software Testing”, John Wiley & Sons, 1979, ISBN 0-471-46912-2

[Splaine01]: Steven Splaine, Stefan P. Jaskiel, “The Web-Testing Handbook”, STQE Publishing, 2001, ISBN 0-970-43630-0

[vanVeenendaal12]: Erik van Veenendaal, “Practical risk-based testing – The PRISMA approach”, UTN Publishers, The Netherlands, ISBN 9789490986070

[Wiegers03]: Karl Wiegers, “Software Requirements 2”, Microsoft Press, 2003, ISBN 0-735-61879-8

[Whittaker03]: James Whittaker, “How to Break Software”, Addison-Wesley, 2003, ISBN 0-201-79619-8

[Whittaker09]: James Whittaker, “Exploratory Software Testing”, Addison-Wesley, 2009, ISBN 0-321-63641-4

8.4 Otras Referencias

Las siguientes referencias apuntan a información disponible en Internet y en otras fuentes. A pesar de que estas referencias fueron consultadas en el momento de la publicación de este programa de estudio de Nivel Avanzado, el ISTQB no se responsabiliza de la disponibilidad de tales referencias.

- Capítulo 3
  - Czerwonka, Jacek: www.pairwise.org
  - Taxonomía de bugs: www.testingeducation.org/a/bsct2.pdf
  - Ejemplo de taxonomía de “bug” basado en el trabajo de Boris Beizer: inet.uni2.dk/~vinter/bugtaxst.doc
  - Buena panorámica de varias taxonomías: testingeducation.org/a/bugtax.pdf
  - Pruebas heurísticas basadas en riesgos de James Bach
  - De “Exploratory & Risk-Based Testing (2004) www.testingeducation.org“
  - Exploring Exploratory Testing , Cem Kaner and Andy Tikam , 2003
  - Pettichord, Bret, “An Exploratory Testing Workshop Report”, www.testingcraft.com/exploratorypettichord
- Capítulo 4
  - www.testingstandards.co.uk

9. Índice Terminológico