Unidad 4
Técnicas de Prueba

1. Categorías de Técnicas de Prueba

El objetivo de una técnica de prueba, incluidas las que se analizan en esta sección, es ayudar a identificar las condiciones de prueba, los casos de prueba y los datos de prueba.

1.1 Elección de Técnicas de Prueba

La elección de qué técnicas de prueba se van a utilizar depende de una serie de factores, entre los que se incluyen los siguientes:

- Tipo de componente o sistema.
- Complejidad del componente o del sistema.
- Estándares de regulación.
- Requisitos del cliente o contractuales.
- Niveles de riesgo.
- Clases de riesgo.
- Objetivos de prueba.
- Documentación disponible.
- Conocimientos y competencias del probador.
- Herramientas disponibles.
- Tiempo y presupuesto.
- Modelo de ciclo de vida de desarrollo de software.
- Uso previsto del software.
- Experiencia previa en el uso de las técnicas de prueba en el componente o sistema que se va a probar.
- Los tipos de defectos esperados en el componente o sistema.

Algunas técnicas son más adecuadas para ciertas situaciones y niveles de prueba; otras se pueden aplicar en todos los niveles de prueba. En general, cuando se crean casos de prueba, los probadores utilizan una combinación de técnicas de prueba para lograr los mejores resultados a partir del esfuerzo de prueba.

El uso de técnicas de prueba en el análisis de la prueba, el diseño de prueba y las actividades de implementación de la prueba puede variar desde muy informal (poca o ninguna documentación) hasta muy formal. El nivel de formalidad adecuado depende del contexto de la prueba, incluyendo la madurez de los procesos de prueba y desarrollo, las limitaciones de tiempo, los requisitos de seguridad o normativos, los conocimientos y competencias de las personas involucradas y el modelo de ciclo de vida de desarrollo de

software que se sigue.

1.2 Categorías de Técnicas de Prueba y sus Características

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

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

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

Las técnicas de prueba basadas en la experiencia aprovechan la experiencia de desarrolladores, probadores y usuarios para diseñar, implementar y ejecutar pruebas. A menudo, estas técnicas se combinan con técnicas de prueba de caja negra y caja blanca.

Entre las características comunes de las técnicas de prueba de la caja negra se incluyen las siguientes:

- Las condiciones de prueba, casos de prueba y datos de prueba se deducen de una base de prueba que puede incluir requisitos de software, especificaciones, casos de uso e historias de usuario.
- Los casos de prueba se pueden utilizar para detectar diferencias[42] entre los requisitos y la implementación de los requisitos, así como desviaciones con respecto a los requisitos.
- La cobertura se mide en función de los elementos probados en la base de prueba y de la técnica aplicada a la base de prueba.

Entre las características comunes de las técnicas de prueba de caja blanca se incluyen las siguientes:

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

Entre las características comunes de las técnicas de prueba basadas en la experiencia se incluyen las siguientes:

- Las condiciones de prueba, los casos de prueba y los datos de prueba se deducen de una base de prueba que puede incluir el conocimiento y la experiencia de probadores, desarrolladores, usuarios y otros implicados.

Este conocimiento y experiencia incluye el uso esperado del software, su entorno, los posibles defectos y la distribución de dichos defectos.

El estándar internacional (ISO/IEC/IEEE 29119-4) contiene descripciones de técnicas de prueba y sus correspondientes medidas de cobertura.

2. Técnicas de prueba de caja negra

2.1 Partición de Equivalencia

La partición de equivalencia divide los datos en particiones (también conocidas como clases de equivalencia) de tal manera que se espera que todos los miembros de una partición dada sean procesados de la misma manera. Existen particiones de equivalencia tanto paravalores válidos como no válidos.

- Los valores válidos son los valores que debe aceptar el componente o el sistema. Una partición de equivalencia que contiene valores válidos se llama "partición de equivalencia válida".
- Los valores no válidos son valores que deben ser rechazados por el componente o sistema. Una partición de equivalencia que contiene valores no válidos se llama "partición de equivalencia no válida".
- Las particiones pueden identificarse para cualquier elemento de datos relacionado con el objeto de prueba, incluyendo entradas, salidas, valores internos, valores relacionados con el tiempo (por ejemplo, antes o después de un evento) y para parámetros de interfaz (por ejemplo, componentes integrados que se están probando durante la prueba de integración).
- Cualquier partición se puede dividir en subparticiones si fuera necesario.
- Cada valor debe pertenecer a una y sólo a una partición de equivalencia.
- Cuando se utilizan particiones de equivalencia no válidas en casos de prueba, deben probarse individualmente, es decir, no combinarse con otras particiones de equivalencia no válidas, para garantizar que no se produzca un enmascaramiento de los fallos. Los fallos se pueden enmascarar cuando se producen varios fallos al mismo tiempo, pero sólo uno de ellos es visible, lo que hace que los otros fallos queden sin detectar.

Para lograr una cobertura del 100% con esta técnica, los casos de prueba deben cubrir todas las particiones identificadas (incluidas las particiones no válidas) utilizando, como mínimo, un valor de cada partición. La cobertura se mide como el número de particiones de equivalencia probadas por al menos un valor, dividido por el número total de particiones de equivalencia identificadas, normalmente expresado como un porcentaje. La partición de equivalencia es aplicable a todos los niveles de prueba.

2.2 Análisis de Valores Frontera

El análisis de valores frontera (AVF) es una extensión de la partición de equivalencia, pero sólo se puede utilizar cuando la partición está ordenada, y consiste en datos numéricos o secuenciales. Los valores mínimo y máximo (o valores inicial y final) de una partición son sus valores frontera (Beizer 1990).

Por ejemplo, suponer que un campo de entrada acepta un único valor entero como entrada, utilizando un teclado numérico para limitar las entradas de modo que las entradas no enteras sean imposibles. El rango válido es de 1 a 5, ambos valores incluidos. Por lo tanto, hay tres particiones de equivalencia: inválida (demasiado baja); válida; inválida (demasiado alta). Para la partición de equivalencia válida, los valores frontera son 1 y 5. Para la partición no válida (demasiado alta), los valores frontera son 6 y 9. Para la partición inválida (demasiado baja), sólo hay un valor frontera, 0, porque se trata de una partición con un solo miembro.

En el ejemplo anterior, se identifican dos valores frontera por frontera. La frontera entre inválido (demasiado bajo) y válido proporciona los valores de prueba 0 y 1. La frontera entre válido e inválido (demasiado alto) proporciona los valores de prueba 5 y 6. Algunas variantes de esta técnica identifican tres valores frontera

por frontera: los valores antes, en y justo después de la frontera. En el ejemplo anterior, utilizando tres puntos para los valores frontera, los valores de la prueba de la frontera inferior son 0, 1 y 2, y los valores de la prueba de la frontera superior son 4, 5 y 6 (Jorgensen 2014).

El comportamiento en las fronteras de las particiones de equivalencia es más probable que sea incorrecto que el comportamiento dentro de las particiones. Es importante recordar que tanto las fronteras especificadas como las implementadas pueden desplazarse a posiciones por encima o por debajo de sus posiciones previstas, pueden omitirse por completo o pueden complementarse con fronteras adicionales no deseadas. El análisis y la prueba del valor frontera revelarán casi todos estos defectos forzando al software a mostrar comportamientos de una partición distinta a la que debería pertenecer el valor frontera.

El análisis de valores frontera se puede aplicar en todos los niveles de prueba. Esta técnica se utiliza generalmente para probar los requisitos que requieren un rango de números (incluyendo fechas y horas). La cobertura de frontera para una partición se mide como el número de valores frontera probados, dividido por el número total de valores frontera de prueba identificados, normalmente expresado como un porcentaje.

2.3 Prueba de Tabla de Decisión

Las técnicas de prueba combinatorias son útiles para probar la implementación de requisitos de sistema que especifican cómo diferentes combinaciones de condiciones generan diferentes resultados. Un enfoque para este tipo de prueba es la prueba de tabla de decisión.

Las tablas de decisión son una buena manera de documentar reglas de negocio complejas que un sistema debe implementar. Al crear tablas de decisión, el probador identifica las condiciones (a menudo entradas) y las acciones resultantes (a menudo salidas) del sistema. Éstas conforman las filas de la tabla, generalmente con las condiciones en la parte superior y las acciones en la parte inferior. Cada columna corresponde a una regla de decisión que define una combinación única de condiciones que resulta en la ejecución de las acciones asociadas a esa regla. Los valores de las condiciones y acciones, normalmente se muestran como valores booleanos (verdadero o falso) o valores discretos (por ejemplo, rojo, verde, azul), pero también pueden ser números o intervalos de números. Estos diferentes tipos de condiciones y acciones pueden estar juntos en la misma tabla.

La notación habitual en las tablas de decisión es la siguiente:

Para las condiciones:

- S[43] significa que la condición es verdadera (también se puede mostrar como V[44] o 1).
- N[45] significa que la condición es falsa (también se puede mostrar como F[46] o 0).
- Un guion "-" significa que el valor de la condición no importa (también puede mostrarse como N/A[47]).

Para las acciones:

- X significa que la acción debe ocurrir (también puede mostrarse como S o V o 1).
- En blanco significa que la acción no debe ocurrir (también puede mostrarse como - o N o F o 0).

Una tabla de decisión completa[48] tiene suficientes columnas para cubrir cada combinación de condiciones. La tabla se puede colapsar borrando las columnas que contienen combinaciones de condiciones imposibles, las columnas que contienen combinaciones de condiciones posibles pero no factibles y las columnas que prueban combinaciones de condiciones que no afectan al resultado.

La cobertura estándar mínima habitual para la prueba de tabla de decisión es tener al menos un caso de prueba por regla de decisión en la tabla. Esto implica, normalmente, cubrir todas las combinaciones de condiciones. La cobertura se mide como el número de reglas de decisión probadas por, al menos, un caso de prueba, dividido por el número total de reglas de decisión, normalmente expresado como un porcentaje.

La fortaleza de la prueba de tabla de decisión es que ayuda a identificar todas las combinaciones importantes de condiciones, algunas de las cuales, de otra manera, podrían ser ignoradas. También ayuda a encontrar cualquier desfase[49] en los requisitos. Puede aplicarse a todas las situaciones en las que el comportamiento del software depende de una combinación de condiciones, en cualquier nivel de prueba.

2.4 Prueba de Transición de Estado

Los componentes o sistemas pueden responder de forma diferente a un evento dependiendo de las condiciones del momento o de su historia previa (por ejemplo, los eventos que han ocurrido desde que se inicializó el sistema). La historia previa puede resumirse utilizando el concepto de estado. Un diagrama de transición de estado muestra los posibles estados del software, así como la forma en que el software entra, sale y realiza las transiciones entre estados. Una transición se inicia con un evento (por ejemplo, la entrada de un valor por parte del usuario en un campo). El evento resulta en una transición. Si el mismo evento puede resultar en dos o más transiciones diferentes desde el mismo estado, ese evento puede estar condicionado por una condición de guarda[50]. El cambio de estado puede provocar que el software tome una acción (por ejemplo, emitir el resultado de un cálculo o un mensaje de error).

Una tabla de transición de estado muestra todas las transiciones válidas y las transiciones potencialmente inválidas entre estados, así como los eventos, las condiciones de guarda y las acciones resultantes para las transiciones válidas. Los diagramas de transición de estado, normalmente, sólo muestran las transiciones válidas y excluyen las transiciones no válidas.

Las pruebas pueden ser diseñadas para cubrir una secuencia de estados típica, para practicar todos los estados, para practicar cada transición, para practicar secuencias específicas de transiciones, o para probar transiciones inválidas.

La prueba de transición de estado se utiliza para aplicaciones basadas en menús y es extensamente utilizada en la industria del software embebido. La técnica también es adecuada para modelar un escenario de negocio con estados específicos o para probar la navegación en pantalla. El concepto de estado es abstracto: puede representar unas pocas líneas de código o todo un proceso de negocio.

La cobertura se mide, habitualmente, como el número de estados o transiciones identificados probados, dividido por el número total de estados o transiciones identificados en el objeto de prueba, normalmente expresado como un porcentaje.

2.5 Prueba de Caso de Uso

Las pruebas se pueden obtener[51] a partir de casos de uso, que son una forma específica de diseñar interacciones con elementos software, incorporando requisitos para las funciones del software representadas por los casos de uso. Los casos de uso están asociados con actores (usuarios humanos, hardware externo u otros componentes o sistemas) y sujetos (el componente o sistema al que se aplica el caso de uso).

Cada caso de uso especifica algún comportamiento que un sujeto puede realizar en colaboración con uno o más actores (UML 2.5.1 2017). Un caso de uso puede describirse mediante interacciones y actividades, así como mediante precondiciones, poscondiciones y lenguaje natural cuando resulte adecuado. Las interacciones entre los actores y el sujeto pueden resultar en cambios en el estado del sujeto. Las interacciones pueden representarse gráficamente mediante flujos de trabajo, diagramas de actividad o

modelos de procesos de negocio.

Un caso de uso puede incluir posibles variaciones de su comportamiento básico, incluyendo el tratamiento de un comportamiento excepcional y de errores (respuesta del sistema y recuperación de errores de programación, de aplicación y de comunicación, por ejemplo, resultando en un mensaje de error). Las pruebas están diseñadas para practicar las conductas definidas (básicas, excepcionales o alternativas, y tratamiento de errores). La cobertura se puede medir por el porcentaje de comportamientos de casos de uso probados dividido por el número total de comportamientos de casos de uso, normalmente expresado como un porcentaje.

3. Técnicas de prueba de Caja Blanca

Las técnicas de prueba de caja blanca se basan en la estructura interna del objeto de prueba. Las técnicas de prueba de caja blanca se pueden utilizar en todos los niveles de prueba, pero las dos técnicas relacionadas con el código que se discuten en esta sección se utilizan con mayor frecuencia en el nivel de prueba de componente. Hay técnicas más avanzadas que se utilizan en algunos entornos de seguridad crítica, de misión crítica o de alta integridad para lograr una cobertura más completa, pero no se tratan en este documento.

3.1 Prueba y Cobertura de Sentencia

La prueba de sentencia practica las sentencias ejecutables en el código. La cobertura se mide como el número de sentencias ejecutadas por las pruebas dividido por el número total de sentencias ejecutables en el objeto de prueba, normalmente expresado como un porcentaje.

3.2 Prueba y Cobertura de Decisión

La prueba de decisión practica las decisiones en el código y prueba el código que se ejecuta basado en los resultados de la decisión. Para ello, los casos de prueba siguen los flujos de control que se producen desde un punto de decisión (por ejemplo, para una declaración IF, uno para el resultado verdadero y otro para el resultado falso; para una declaración CASE, se necesitarían casos de prueba para todos los resultados posibles, incluido el resultado por defecto).

La cobertura se mide como el número de resultados de decisión ejecutados por las pruebas dividido por el número total de resultados de decisión en el objeto de prueba, normalmente expresado como un porcentaje.

3.3 El Valor de la Prueba de Sentencia y Decisión

Cuando se logra una cobertura del 100% de sentencia, se asegura de que todas las sentencias ejecutables del código se han probado al menos una vez, pero no asegura que se haya probado toda la lógica de decisión. De las dos técnicas de caja blanca discutidas en este programa, la prueba de sentencia puede proporcionar menos cobertura que la prueba de decisión.

Cuando se alcanza el 100% de cobertura de decisión, se ejecutan todos los resultados de decisión, lo que incluye probar el resultado verdadero y también el resultado falso, incluso cuando no hay una sentencia falsa explícita (por ejemplo, en el caso de una sentencia IF sin un ELSE en el código). La cobertura de sentencia ayuda a encontrar defectos en el código que no fueron practicados por otras pruebas. La cobertura de decisión ayuda a encontrar defectos en el código donde otras pruebas no han tenido ambos

resultados, verdadero y falso.

Lograr una cobertura del 100% de decisión garantiza una cobertura del 100% de sentencia (pero no al revés).

4. Tecnicas de prueba basadas en la Experiencia

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

En las siguientes secciones se abordan las técnicas basadas en la experiencia que se utilizan con frecuencia.

4.1 Predicción de Errores

La predicción de errores es una técnica utilizada para anticipar la ocurrencia de equivocaciones, defectos y fallos, basada en el conocimiento del probador, incluido:

- Cómo ha funcionado la aplicación en el pasado.
- Qué tipo de equivocaciones tienden a cometer los desarrolladores.
- Fallos que se han producido en otras aplicaciones.

Un enfoque metódico de la técnica de predicción de errores es crear una lista de posibles equivocaciones, defectos y fallos, y diseñar pruebas que expongan esos fallos y los defectos que los causaron. Estas listas de equivocaciones, defectos y fallos se pueden crear en base a la experiencia, los datos de defectos y fallos, o a partir del conocimiento común de por qué falla el software.

4.2 Prueba Exploratoria

En la prueba exploratoria, se diseñan, ejecutan, registran y evalúan de forma dinámica pruebas informales (no predefinidas) durante la ejecución de la prueba. Los resultados de la prueba se utilizan para aprender más sobre el componente o sistema, y para crear pruebas para las áreas que pueden necesitar ser probadas con mayor intensidad.

A veces se realiza la prueba exploratoria utilizando la prueba basada en sesiones para estructurar la actividad. En la prueba basada en sesiones, la prueba exploratoria se lleva a cabo dentro de un período de tiempo definido, y el probador utiliza un contrato de prueba que contiene los objetivos de la prueba para guiar la prueba. El probador puede usar hojas de sesión de prueba[52] para documentar los pasos seguidos y los descubrimientos realizados.

La prueba exploratoria es más útil cuando las especificaciones son escasas o inadecuadas o cuando hay una presión significativa con respecto al tiempo para la prueba. La prueba exploratoria también es útil para complementar otras técnicas de prueba más formales.

La prueba exploratoria está fuertemente asociada con las estrategias de prueba reactivas. La prueba exploratoria puede incorporar el uso de otras técnicas de caja negra, caja blanca y basadas en la

experiencia.

4.3 Prueba Basada en Listas de Comprobación

En la prueba basada en listas de comprobación, los probadores diseñan, implementan y ejecutan pruebas para cubrir las condiciones de prueba que se encuentran en una lista de comprobación. Como parte del análisis, los probadores crean una nueva lista de comprobación o amplían una ya existente, pero los probadores también pueden utilizar una lista de comprobación ya existente sin modificar. Estas listas de comprobación pueden elaborarse basándose en la experiencia, el conocimiento de lo que es importante para el usuario o la comprensión de por qué y cómo falla el software.

Se pueden crear listas de comprobación para dar soporte a varios tipos de prueba, incluyendo pruebas funcionales y no funcionales. A falta de casos de prueba detallados, las pruebas basadas en listas de comprobación pueden proporcionar directrices y un cierto grado de consistencia. Dado que se trata de listas de alto nivel, es probable que se produzca cierta variabilidad en las pruebas reales, lo que puede dar lugar a una mayor cobertura pero a una menor repetibilidad.