

Fig. 3.4 Proceso de revisión genérico


proceso de revisión específico a una situación determinada. Si la revisión requerida es más formal, entonces habrá más tareas o elementos del proceso. En la Fig. 3.4 se muestra un proceso de revisión genérico descrito en ISO/IEC 20246 .

El tamaño de muchos productos de trabajo significa que pueden ser demasiado grandes para ser cubiertos por una sola revisión. En tales casos, el proceso de revisión generalmente se aplica varias veces a las partes individuales que componen el producto del trabajo.

El programa de estudios del nivel básico describe cinco tipos genéricos de actividades que se muestran en la figura 3.4 y que pueden ocurrir en el proceso de revisión del producto del trabajo. Los comentamos a continuación.

Planificación

La planificación define el alcance del trabajo que cubrirá la revisión. La planificación establece los límites de la revisión respondiendo preguntas como quién, qué, dónde, cuándo y por qué. Se define el propósito de la revisión (respondiendo a la pregunta "por qué"), así como qué documentos serán objeto de la revisión. Se definen las características de calidad a revisar (respondiendo a la pregunta "qué"). Se estima la cantidad de trabajo necesario para realizar la revisión y se definen el marco de tiempo y la ubicación de ciertas fases del proceso de revisión, como la reunión de revisión (respondiendo las preguntas de "dónde" y "cuándo"). Con base en el propósito de la revisión, se define el tipo de revisión, junto con los roles, actividades y listas de verificación que se utilizarán (si es necesario). Se seleccionan las personas que participarán en la revisión y se asignan roles (respuesta a la pregunta "quién").

Para revisiones formales  como la inspección, se definen criterios formales de entrada y salida, y al final de esta fase también se verifica que se cumplan los criterios de entrada y que la revisión pueda pasar a la siguiente fase.

Inicio de la revisión

Como parte del inicio de la revisión, el producto del trabajo que se va a revisar se envía a los participantes de la revisión (seleccionados en la fase de planificación), junto con todos los demás materiales necesarios, por ejemplo, listas de verificación, declaraciones de procedimiento e informe de defectos. plantillas. Todos los materiales deben distribuirse lo antes posible para que los revisores tengan tiempo suficiente para completar la revisión.

Si es necesario, se dan explicaciones a los participantes sobre cómo se llevará a cabo la revisión y cuál es su papel. Si los participantes tienen alguna pregunta, se brindan respuestas y explicaciones durante la fase de inicio de la revisión (antes de que se lleve a cabo la revisión real del producto del trabajo) para que todos estén conscientes de lo que se supone que deben hacer y conozcan el cronograma de estas actividades.

Es posible organizar una capacitación de revisión durante esta fase. Esto tiene sentido cuando los participantes no tienen experiencia en revisiones y desea que el proceso de revisión se desarrolle sin problemas y de manera eficiente. Para revisiones formales, también se puede celebrar una reunión inicial para explicar a los participantes individuales el alcance de la revisión y sus roles y responsabilidades individuales como revisores.

El propósito de la fase de inicio de la revisión es garantizar que todos los involucrados en la revisión estén preparados y listos para comenzar la revisión.

Revisión individual (es decir, preparación individual)

Esta es la fase central y esencial de cualquier revisión. En esta fase se realizan actividades sustantivas, es decir, se lleva a cabo la revisión real del producto del trabajo por parte de los revisores, utilizando técnicas de revisión específicas (ver Sección 3.2.6). Como parte de estas actividades, se revisa todo o parte del producto del trabajo (la parte que se va a revisar debe determinarse durante la planificación y explicarse cuidadosamente a los participantes en la fase de inicio de la revisión). Los revisores registran cualquier comentario, pregunta, recomendación, inquietud y observación relevante realizada durante la revisión del producto del trabajo.

De todas las fases de revisión, la fase de revisión individual detecta el mayor porcentaje de problemas. Los problemas identificados generalmente se documentan en un registro de problemas, que a menudo está respaldado por una herramienta de soporte de revisión o gestión de defectos.

Según la norma ISO/IEC 20246 [6], este paso del proceso es opcional⁷ y no puede realizarse para ciertos tipos de revisión, como recorridos o revisiones informales. Sin embargo, muchos autores adoptan la posición de que es la actividad más importante de todo el proceso de revisión.

Comunicación y análisis Si no se

programa una reunión de revisión, los problemas encontrados se comunican directamente a las personas (por ejemplo, el autor del producto de trabajo que se está revisando), junto con un análisis de esos problemas. Si la reunión de revisión está incluida en el plan de revisión, los problemas se informan directamente a los participantes de la reunión o, lo que probablemente sea mejor, se envían colectivamente a todos los participantes antes de la reunión para que los revisen con anticipación, de modo que la reunión de revisión funcionará de manera más eficiente.

La reunión incluye un análisis de los problemas (anomalías) informados por los revisores. Dado que no todas las anomalías necesariamente resultan ser un defecto, todos los hallazgos deben revisarse cuidadosamente en la reunión para decidir si existe un problema real (defecto) o simplemente un problema aparente (falso positivo). Si la anomalía resulta ser un defecto, se designan los responsables de solucionarla y se definen parámetros como estado, prioridad o gravedad. Estas acciones se realizan en una revisión.

⁷ En tales situaciones, la detección de defectos puede tener lugar, por ejemplo, durante la llamada reunión de revisión que se analiza en el siguiente párrafo.

reunión o (si no existe dicha reunión) individualmente. Los estados más utilizados para anomalías reportadas son:

- Problema rechazado •
- Problema registrado sin tomar ninguna medida • Problema a resolver por el autor del producto de trabajo • Problema actualizado como resultado de un análisis adicional •
- Problema asignado a una parte interesada externa

Esta fase también evalúa y documenta el nivel de las características de calidad que se definieron en la fase de planificación como las que se están revisando. Finalmente, las conclusiones de la revisión se evalúan según los criterios de salida para decidir qué hacer con el producto del trabajo revisado. Ejemplos de decisiones incluyen:

- Aceptación del producto del trabajo (normalmente cuando no hay problemas o sólo un pequeño se detectan varios problemas insignificantes).
- Actualizar el producto de trabajo en base a los problemas identificados (decisión típica). • El producto del trabajo debe revisarse nuevamente (generalmente debido a la gran cantidad de problemas y el gran alcance de los cambios).
- Rechazo del producto del trabajo (el tipo de decisión más rara, pero también puede ocurrir).

Reparación e informes

Esta es la etapa final de la revisión. Crea informes de defectos detectados que requieren cambios. Se espera que el autor del producto de trabajo revisado lleve a cabo la eliminación de defectos durante esta fase. Esto se hace informando a las personas relevantes o al equipo relevante de los defectos detectados en el producto de trabajo bajo revisión. Finalmente, cuando se confirman los cambios, se crea un informe de revisión.

Para tipos de revisión más formales, además, se recopilan y utilizan varios tipos de medidas para probar la eficacia de la revisión. También se compara con los criterios de salida y el producto del trabajo se acepta una vez que se determina que se han cumplido los criterios de salida.

Los resultados de una revisión del producto de trabajo pueden variar según el tipo de revisión y el grado de formalización (ver Apartado [3.2.4](#)).

Ejemplo Un equipo ha decidido realizar una revisión del llamado árbol de habilidades en un juego de rol en línea para computadora que están produciendo. El líder de la prueba selecciona a cinco personas como participantes en la revisión: un desarrollador, un evaluador y tres personas externas al equipo, que actúan como partes interesadas externas (jugadores). Se decide que el tipo de revisión realizada será una revisión informal individual.

El líder del equipo distribuye a los participantes un documento que describe el árbol de habilidades, una lista de verificación de las características que se deben verificar y una plantilla de registro de problemas, que se muestra en la Tabla [3.2](#).

Cada participante, en un momento predeterminado, revisa el documento y completa el formulario de registro de problemas y luego lo envía al líder del equipo. El líder del equipo fusiona los formularios en uno, elimina los defectos redundantes (posiblemente marcando que fueron encontrados por más de una persona) y envía la lista de problemas así creados al

Tabla 3.2 Plantilla de registro de problemas

No. Página/línea	Categoría de defecto	Tipo de defecto de gravedad	Fuente del defecto	Descripción, comentarios
...
Defect categories: Missing, Defect, Redundancy				

Tabla 3.3 Un registro de problemas completado por uno de los revisores

No. Página/línea	Categoría de defecto	Gravedad	Tipo de defecto	Descripción, comentarios	
1	Figura 1	Defecto	Pequeño	Sintaxis	"Iluminación eléctrica" en lugar de "Rayo eléctrico"
2	Figura 1	Defecto	Grande	Lógica	El elemento que sigue a "Eléctrico Rayo" debería ser la habilidad "Tormenta eléctrica", no "Tormenta de fuego".
3	Figura 1	Desaparecido	Grande	Lógica	La habilidad "cegar al oponente" falta, la introducción de cuál al árbol de habilidades se acordó en la reunión del 4 de febrero

desarrolladores, al mismo tiempo transformando los defectos detectados en informes de defectos y los registra en la herramienta de gestión de defectos.

En la Tabla 3.3 se muestra un ejemplo de los defectos informados por uno de los revisores . Dado que el revisor no utilizó la lista de verificación, la columna "Fuente del defecto" no fue necesario, por lo que esta columna se eliminó del registro de problemas.

Los autores corrigen los defectos encontrados y los registran en la herramienta de gestión de defectos. El líder de revisión verifica que se hayan solucionado todos los defectos. El líder de revisión entonces crea un informe de revisión resumido que contiene información sobre:

- Número de personas que participan en la revisión
- Duración de la revisión
- Tamaño del documento que se está viendo (por ejemplo, número de líneas de código, número de páginas del documento)
- Número de defectos encontrados, desglosados por gravedad (grandes/pequeños)

Aquí es donde termina el proceso de revisión.

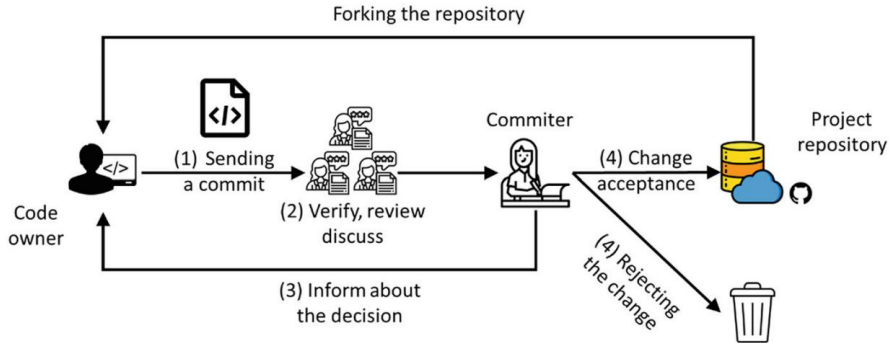


Fig. 3.5 Proceso de revisión de código moderno

Efectividad de las reuniones de revisión

El plan de estudios de Foundation Level dice que la reunión de revisión es uno de los pasos del proceso de revisión. Si analizamos históricamente cómo se han organizado las revisiones, podemos ver que hasta hace algún tiempo, las revisiones se centraban en reuniones de revisión, que eran la actividad principal y sustantiva del proceso de revisión. En los últimos doce años se ha estudiado el problema de la necesidad y eficacia de estas reuniones. Estos estudios han concluido que, en general, estas reuniones no aumentan significativamente la eficacia de la detección de defectos (que suele ser el objetivo más importante de las revisiones). Las reuniones de revisión son costosas, por lo que no parecen agregar ningún valor particular. Además, resulta que si hay demasiadas personas en una reunión (más de cinco), la sobrecarga de comunicación comienza a tener un impacto negativo en el aprendizaje y mejora de habilidades de los asistentes.

Por otro lado, las reuniones de revisión son una excelente oportunidad para hablar e intercambiar experiencias, compartir conocimientos o lograr consensos. Los estudios también demuestran que estas reuniones ayudan a reducir el número de falsos positivos (informes de anomalías que resultan no ser defectos). Así, el valor añadido puede revelarse no necesariamente en el número de defectos detectados, sino en la mejora de todo el proceso de desarrollo y en la mejora de las habilidades de los miembros del equipo [32].

Revisión de código moderno (MCR)

Hoy en día, muchas empresas utilizan un proceso de revisión de código conocido como Revisión de código moderno (MCR). En la figura 3.5 se muestra un diagrama de este proceso. MCR es una técnica eficaz de control de calidad que puede verificar la calidad del software y la satisfacción del cliente identificando defectos, mejorando el código y acelerando el proceso de desarrollo. Es un proceso de revisión asincrónico y liviano respaldado por herramientas de revisión como Gerrit. Es una versión ligera del proceso de inspección de Fagan (ver Sección 3.2.4) y ha evolucionado como una práctica para el desarrollo de software industrial y de código abierto [33].

3.2.3 Funciones y responsabilidades en las revisiones

En una revisión típica, se distinguen los siguientes roles (la descripción de los roles y la lista de responsabilidades se han tomado del programa de estudios; al final, damos algunas notas adicionales sobre algunos de los roles):

Gerente •

Es responsable de programar la revisión • Decide realizar una revisión

- Designa personal y establece un presupuesto y un calendario.
- Supervisa la rentabilidad de la revisión de forma continua. • Ejecuta decisiones de control en caso de resultados insatisfactorios.

Autor •

Crea el producto de trabajo bajo revisión • Elimina defectos en el producto de trabajo bajo revisión (si es necesario)

El autor es la persona responsable de preparar los materiales para revisión, aunque formalmente los materiales pueden ser distribuidos por el líder de revisión. Si es necesario, el autor puede proporcionar a los revisores explicaciones técnicas del producto del trabajo que se está revisando. Es muy importante que los autores comprendan y acepten las críticas profesionales de los revisores sobre el producto de su trabajo y que no defiendan ni nieguen los comentarios de los revisores. Después de todo, el objetivo de la reseña suele ser descubrir tantos problemas como sea posible, y la crítica no está dirigida al autor (porque todo el mundo es falible), sino al producto.

Sin embargo, puede suceder que el autor no comprenda el comentario del revisor y el autor no comprenda cuál es el problema y, por lo tanto, no sepa cómo solucionarlo. Entonces puede ser necesario comunicarse entre el autor y el revisor para explicar las inquietudes del revisor con más detalle.

El autor también puede evaluar el trabajo de los revisores en términos del valor de los comentarios que hacen. Este tipo de retroalimentación se puede utilizar para planificar revisiones futuras en la organización.

Moderador (también conocido como facilitador) •

Garantiza el buen funcionamiento de las reuniones de revisión (si se llevan a cabo) • Actúa como mediador si es necesario conciliar diferentes puntos de vista • Garantiza que se cree una atmósfera segura de confianza y respeto mutuos en la revisión reunión

Escriba (también conocido como registrador)

• Reúne posibles anomalías detectadas e informadas como parte de la revisión individual • Registra nuevos defectos potenciales encontrados durante la reunión de revisión, así como decisiones tomadas en la reunión (si se lleva a cabo)

El escribano debe desempeñar un papel "transparente" durante la reunión de revisión, es decir, debe ser "invisible" para los demás participantes. Su función es aliviar al otro.

participantes de la tarea de registrar cualquier comentario realizado durante la reunión. Una reunión de revisión requiere la máxima concentración por parte de los revisores y del autor, y trabajarán de manera más efectiva cuando no tengan que distraerse cada vez que se encuentre un nuevo problema y sea necesario escribirlo.

Revisor •

Realiza una revisión, identificando defectos potenciales en el producto de trabajo bajo revisión. • Puede ser un experto en la materia, una persona que trabaja en el proyecto, una parte interesada en el producto de trabajo y/o una persona con experiencia técnica o comercial específica.

- Puede representar diferentes puntos de vista (por ejemplo, el punto de vista de un evaluador, desarrollador, usuario, operador, analista de negocios, especialista en usabilidad)

Los revisores desempeñan un papel clave en el proceso de revisión, ya que encuentran problemas en el producto del trabajo que se está revisando, y este suele ser el objetivo principal de las revisiones. A los revisores se les debe dar tiempo suficiente para prepararse e informar los problemas que encuentren. Los revisores deben dirigir sus comentarios al producto, no al autor.

Líder de revisión •

Tiene la responsabilidad general del proceso de revisión • Decide quién participará en la revisión, determina el lugar y la fecha de la revisión y es responsable de organizar las reuniones de revisión

Roles versus personas

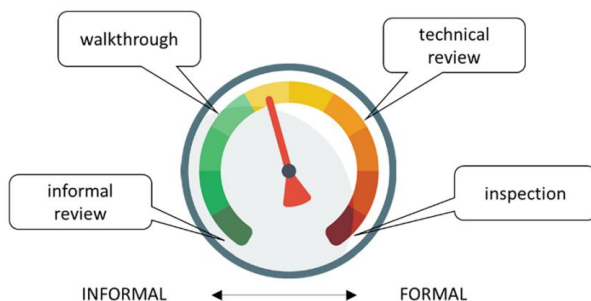
Para algunos tipos de revisiones, una persona puede desempeñar varios roles y, según el tipo de revisión, las actividades asociadas con cada rol también pueden variar. Además, con la llegada de herramientas para ayudar en el proceso de revisión (y en particular el registro de defectos, puntos abiertos y decisiones), a menudo no es necesario nombrar un escribano. Tampoco será necesario un escribano si no se planea una reunión de revisión.

3.2.4 Tipos de revisión

Hay muchos tipos de revisiones con diferentes niveles de formalidad, desde revisiones informales hasta revisiones altamente formalizadas (ver Fig. 3.6). El nivel de formalidad requerido depende de factores como el modelo SDLC utilizado, la madurez del proceso de desarrollo, la criticidad y complejidad del producto de trabajo que se revisa, cualquier requisito legal o reglamentario y la necesidad de una pista de auditoría.

Elegir el tipo correcto de revisión es fundamental para lograr los objetivos de revisión requeridos. La selección se basa no sólo en los objetivos sino también en factores como las necesidades del proyecto, los recursos disponibles, el tipo de producto del trabajo, los riesgos, el dominio empresarial y la cultura organizacional.

Fig. 3.6 Nivel de formalidad de los diferentes tipos de revisión



El programa de estudios de Foundation Level describe cuatro tipos de revisiones:

- Inspección •
- Revisión técnica •
- Recorrido • Revisión informal

La inspección es una revisión formal. La revisión técnica y el recorrido pueden variar desde muy formalizados hasta bastante informales. El sello distintivo de las revisiones informales es que no siguen un proceso definido y la información obtenida a través de ellas no necesita estar documentada formalmente. Las revisiones formales, en cambio, se realizan de acuerdo con procedimientos documentados y con la participación de un equipo preestablecido, y los resultados deben documentarse obligatoriamente.

Las revisiones se pueden realizar con diversos fines, pero uno de los objetivos principales de cualquier revisión es encontrar defectos. Las reseñas se pueden clasificar según varios atributos. La Tabla 3.4 muestra los tipos de revisión más comunes discutidos en el programa de estudios de Foundation Level, junto con sus atributos correspondientes. A continuación, proporcionaremos información más detallada sobre estos tipos de revisiones.

Orden de revisiones

Un producto de trabajo puede estar sujeto a más de un tipo de revisión, y si se realizan varias revisiones de diferentes tipos, su orden puede variar. Por ejemplo, se puede realizar una revisión informal antes de una revisión técnica para garantizar que el producto del trabajo esté listo para la revisión técnica.

Revisiones por

pares Todos los tipos de revisiones pueden implementarse como revisiones por pares, es decir, realizadas por colegas de un nivel organizacional similar o con responsabilidades o experiencia similares.

Tipos de defectos encontrados durante las revisiones

Los tipos de defectos encontrados en las revisiones varían, dependiendo en particular del producto de trabajo que se revisa. Para ver ejemplos de defectos encontrados en revisiones de diversos productos de trabajo, consulte la Sección. 3.1.3, y para obtener información sobre revisiones formales, consulte [34].

Pasamos ahora a una discusión detallada de los cuatro tipos de revisiones descritas en el programa de estudios. En la Tabla 3.4 se resume una comparación resumida de los tipos de revisión que se analizan a continuación .

Tabla 3.4 Comparación de tipos de revisión

tipo de reseña	Informal revisar	Tutorial	Revisión técnica	Inspección
Objetivos principales	Detectar posibles defectos	Detectar potencial defectos, mejorar calidad, considere alternativas, evaluar el cumplimiento con estándares	Obtener consenso, detectar potencial defectos	Detectar potencial defectos, evaluar el calidad de la producto de trabajo, aumentar la confianza en él, prevenir similares defectos de ocurriendo en el futuro
Adiciones potenciales objetivos nacionales	Generar nuevo ideas, rápidamente resolver sencillo problemas	Intercambiar información, capacitar a los participantes, llegar consenso	evaluar la calidad de un producto de trabajo, aumentar la confianza en él, generar nuevas ideas, motivar a los autores a mejorar el futuro productos del trabajo, evaluar alternativas	Motivar a los autores para mejorar el futuro productos de trabajo y El software proceso de desarrollo, crear condiciones para ello, llegar a un consenso
Proceso formal Ninguno		Preparación individual opcional antes de reunión	Preparación individual obligatoria antes de la reunión	Proceso formal basado en reglas y listas de verificación; individuo obligatorio preparación antes reunión
Roles	Se puede ejecutar por un autor, por un autor compañero, o por un grupo de gente	La Junta es generalmente presidido por el autor; el presencia de un se requiere escribano	La reunión debiera ser realizado por un moderador, no por el autor; el presencia de un se requiere escribano	Estrictamente definido; La reunión está dirigida por un moderador, no el autor; escriba es obligatorio; lector opcional role
Documentación de resultados	Opcional	Defecto opcional registros y revisión los informes son creado	En general, defecto registros y revisión se crean informes	En general, defecto registros y revisión se crean informes
Nivel de formalismo	Informal	De informal a formal; puede tomar la forma de escenarios, simulacros, o simulaciones	De informal a formal; revisar reunión opcional	Formal; entrada y Los criterios de salida están en lugar; medidas se recogen que se utilizan para mejorar todo proceso, incluyendo la inspección proceso
Listas de verificación	Opcional	Opcional	Opcional	Usualmente usado

Revisión informal La

revisión informal no sigue ningún proceso establecido o predefinido. El resultado de una revisión informal tampoco está documentado de manera formal. El objetivo principal de una revisión informal es detectar posibles anomalías.

Un ejemplo de revisión informal podría ser una conversación entre dos desarrolladores, cuando uno de ellos le pide ayuda al otro para encontrar un defecto que causa un error de compilación. Otro ejemplo podría ser una conversación entre dos ingenieros en la cocina de la oficina durante el almuerzo sobre algún problema técnico. Después de una revisión informal, a menudo no queda rastro y no se documenta nada. En metodologías ágiles, este es el tipo de revisión más común.

Ejemplos de tipos de revisión informal incluyen:

- Revisión de compañeros
- Revisión de pareja

Tutorial El tutorial

implica la revisión secuencial de un producto de trabajo. Lo lleva a cabo el autor del producto del trabajo y puede incluir varios objetivos diferentes, como evaluar la calidad del producto del trabajo, aumentar la confianza en el producto del trabajo, mejorar la comprensión de los revisores sobre el producto del trabajo, llegar a un consenso y generar nuevas ideas. y motivar a los autores a crear mejores productos y encontrar defectos de manera más efectiva. Los revisores pueden realizar una preparación individual antes de la reunión de recorrido, pero no es obligatorio.

Este tipo de revisión también se suele utilizar cuando el equipo no puede detectar la causa de una falla del software. Luego puede adoptar la forma de los llamados ensayos. Una ejecución de prueba puede implicar una simulación manual de la ejecución del código. Luego, el equipo analiza el código cuidadosamente, línea por línea, para comprender bien cómo funciona y localizar el defecto que causa la falla.

El tutorial lo realiza el autor del código, ya que el autor generalmente puede explicar sobre la marcha qué hace el código (o al menos cuáles eran sus intenciones a este respecto cuando implementaron el código). Cuando las revisiones de código se realizan en reuniones de revisión, generalmente toman la forma de un recorrido.

Revisión técnica Los

objetivos de una revisión técnica, realizada por revisores técnicamente calificados, son lograr consenso y tomar decisiones sobre un problema técnico, pero también detectar defectos potenciales, evaluar la calidad y generar confianza en el producto del trabajo, generar nuevas ideas y motivar. y permitir a los autores mejorar el producto de su trabajo.

Una revisión técnica generalmente toma la forma de un "panel de expertos", es decir, una reunión en la que personas competentes tienen la tarea de tomar alguna decisión técnica o de diseño, generalmente importante y significativa, que tiene un impacto importante en el desarrollo futuro del proyecto. . Por este motivo, es obligatoria la preparación individual antes de la reunión, para que en la propia reunión todos puedan participar de forma informada y competente.

En general, una reunión de revisión en una revisión técnica es opcional. Sin embargo, si se lleva a cabo, debe ser dirigido por un facilitador. Por lo general, la revisión técnica finaliza con un informe, ya que debe dejarse un rastro, especialmente si se tomaron decisiones de diseño importantes durante la revisión.

Inspección

Las inspecciones son uno de los tipos de revisión más formales. Su origen se remonta a la década de 1970, cuando Michael Fagan los introdujo en IBM [35]. La realización de inspecciones de software se considera una de las llamadas mejores prácticas en ingeniería de calidad del software debido a los indudables beneficios que aporta. La alta dirección ha enfatizado sistemáticamente el importante papel de las revisiones por pares (incluidas las inspecciones) como elemento clave para garantizar una alta calidad del producto final [36].

Dado que las inspecciones son el tipo de revisión más formal, siguen el proceso de revisión completo descrito en la Sección. 3.2.2. El objetivo principal es lograr la máxima eficiencia en la búsqueda de defectos. Otros objetivos son evaluar la calidad, generar confianza en el producto del trabajo y motivar y permitir a los autores mejorar sus productos. Una característica especial de las inspecciones es recopilar métricas y utilizarlas para mejorar todo el proceso de desarrollo de software, incluido el proceso de inspección en sí. En las inspecciones, el autor no debe asumir el papel de líder de revisión, moderador o escribano.

Las inspecciones siguen un proceso bien definido basado en reglas y listas de verificación. Los resultados de las inspecciones deben documentarse. Un ejemplo de una lista de verificación utilizada durante una inspección podría ser el siguiente.

Lista de verificación de requisitos

Lo completo

1. ¿Los requisitos especificados se relacionan con la misión del proyecto de manera consistente?
¿manera?
2. ¿Los requisitos incluyen las necesidades clave y críticas del usuario, operador y
¿equipo de apoyo?
3. ¿Es cada requisito una unidad de especificación independiente de los demás o tiene dependencias
claramente definidas?
4. ¿No existen deficiencias en los requisitos?
5. ¿Se distinguen los requisitos necesarios de los requisitos opcionales?

Exactitud

1. ¿Los requisitos no son contradictorios entre sí?
2. ¿Se realiza un seguimiento de los requisitos hasta el diseño y el código?
3. ¿Cada requisito tiene un número único?

Estilo

1. ¿Cada requisito es fácil de entender y está escrito en un lenguaje sencillo?
2. ¿Los requisitos hacen una distinción clara entre editorial y funcional?
¿cambios?
3. ¿Se utilizan de manera coherente y uniforme la nomenclatura y las definiciones de los términos?

Medidas de inspección Aunque

muchas organizaciones utilizan inspecciones, hay pocos datos disponibles públicamente sobre su eficacia. Sin embargo, basándose en la información existente (particularmente del Experimento Nacional de Calidad del Software, realizado en 1992), se pueden extraer las siguientes conclusiones [37]:

- No se rastrea el código fuente hasta los requisitos, lo que resulta en pérdida de control “intelectual”, imprecisión de los procedimientos de verificación y dificultad en la gestión de cambios.
- Las buenas prácticas de escritura de código se aplican de manera no rigurosa y no sistemática, lo que resulta en un alto porcentaje de defectos en lógica, datos, interfaces y funcionalidad.
- Los diseños arquitectónicos de las aplicaciones a menudo se crean ad hoc, lo que reduce drásticamente la comprensibilidad, adaptabilidad y mantenibilidad del producto.
- No se hace un uso significativo de los patrones de diseño y programación existentes.
- La distribución de los tipos de defectos detectados durante la inspección es la siguiente:
 - Defectos de documentación—40,51% –
 - Incumplimiento de estándares—23,20% – Defectos de
 - lógica—7,22% – Defectos
 - funcionales—6,57% – Defectos de
 - sintaxis—4,79% – Defectos de
 - datos—4,62% – Defectos de
 - mantenibilidad—4,09%
- En promedio, un equipo necesita entre 8 y 16 minutos para detectar un defecto y alrededor de 80 minutos para detectar un defecto de alta gravedad (conocido como defecto mayor).
- Aproximadamente 3,2 defectos mayores y 17 defectos menos significativos por cada Se encuentran 1000 líneas de código en el código bajo inspección.
- En promedio, el equipo es capaz de analizar 625 líneas de código en una hora.
- En promedio, el equipo encuentra 4,6 defectos por sesión.
- La relación entre el esfuerzo de preparación para la inspección y el esfuerzo de inspección es 0,58.
- La tasa de detección de defectos oscila entre el 80% y el 90%; esto significa que alrededor del 80% al 90% de todos los defectos detectados se encuentran durante la inspección.
- El retorno de la inversión (ROI calculado como la relación entre ganancias y costos) es 4.1.

Las métricas que se pueden recopilar como parte del proceso de inspección incluyen:

- Tiempo de preparación por defecto
- Tiempo de preparación por defecto mayor
- Número de defectos críticos por 1000 líneas de código
- Número de defectos no críticos por 1000 líneas de código
- Número de líneas de código verificadas en 1 h

(continuado)

- Número de defectos por sesión •

Esfuerzo para preparar versus esfuerzo para
inspeccionar • Número de líneas revisadas en una sesión

El análisis de estas métricas, especialmente a partir de una serie de inspecciones realizadas, permite estimar el coste, el esfuerzo y la eficacia de esta forma de prueba estática y calcular el retorno de la inversión. Estos análisis pueden resultar muy útiles para convencer a los directivos de que utilicen revisiones en el trabajo diario de los equipos, demostrando su eficacia y que dan lugar a una reducción del coste general de producción y mantenimiento del software.

Ejemplo Como parte de las técnicas de prueba estáticas, la organización ABC utiliza dos tipos de revisiones: recorridos e inspecciones. Los tutoriales se utilizan con fines de verificación. Se llevan a cabo para adquirir conocimientos sobre diversos aspectos del producto del trabajo. Un objetivo secundario de los tutoriales es:

- Lograr una visión de producto consistente y unificada dentro de la organización • Obtener consenso entre las partes interesadas sobre características específicas del producto • Obtener acuerdo sobre las técnicas de ingeniería que se utilizarán • Determinar la integridad y corrección de las capacidades y características de el software que se está desarrollando

Las revisiones son realizadas por los autores de cada producto de trabajo. Dentro de cada actividad del ciclo de desarrollo, se pueden realizar varias revisiones del mismo producto de trabajo. La única cantidad medida es el número de recorridos realizados.

Se realizan inspecciones para cumplir con los requisitos del equipo responsable de la gestión de calidad. Este equipo verifica que el producto del trabajo cumpla con los estándares aplicables que la organización debe seguir. Al final de cada actividad del ciclo de desarrollo se llevan a cabo inspecciones formales, las llamadas puertas de calidad, durante las cuales se verifican criterios de salida específicos para una fase de desarrollo determinada.

Dado que la inspección es un proceso formal, sigue un proceso bien definido que se muestra en la figura 3.7.

La inspección verifica estricta y formalmente el producto del trabajo para:

- Integridad •
- Corrección •
- Coherencia •
- Estilo •
- Normas de construcción

La inspección la lleva a cabo el moderador y, además, en el proceso participan un escriba, de dos a cinco revisores y el autor. La realización de una inspección se trata como una verificación formal de los criterios de salida de una fase. Durante la inspección, se toman mediciones del producto y del proceso, y se informa toda la sesión en

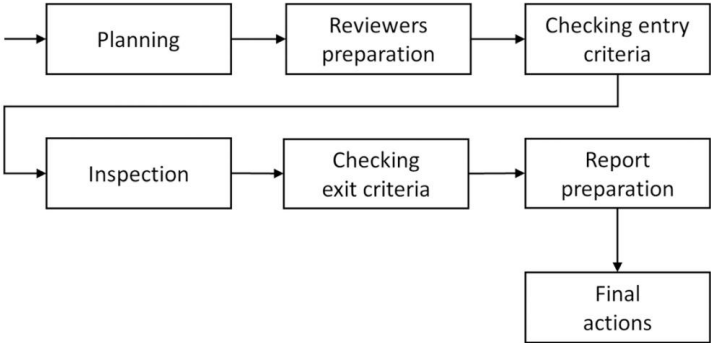


Fig. 3.7 Proceso de inspección en una organización ABC

Tabla 3.5 Comparación de recorridos e inspecciones utilizados en la organización ABC

Categoría	Tutorial	Inspección
Objetivo general	Hacer el trabajo correcto	hacer bien el trabajo
Específico objetivo	Educación, comprensión, consenso	Detección de defectos, cumplimiento comprobación
Desencadenar	solicitud del autor	Criterios de salida de fase
Medición	Instancias de tutorial	Mediciones de productos y procesos.

plantillas de informes definidas específicamente para este fin. Defectos encontrados durante el Las inspecciones se rastrean en la herramienta de gestión de defectos hasta que se cierran.

La Tabla 3.5 resume las diferencias entre recorridos e inspecciones.
Para obtener más información sobre revisiones, consulte [34, 38, 39].

Concluiremos esta sección con dos ejemplos de lo que es una revisión de muestra del cómo podría verse la arquitectura del sistema.

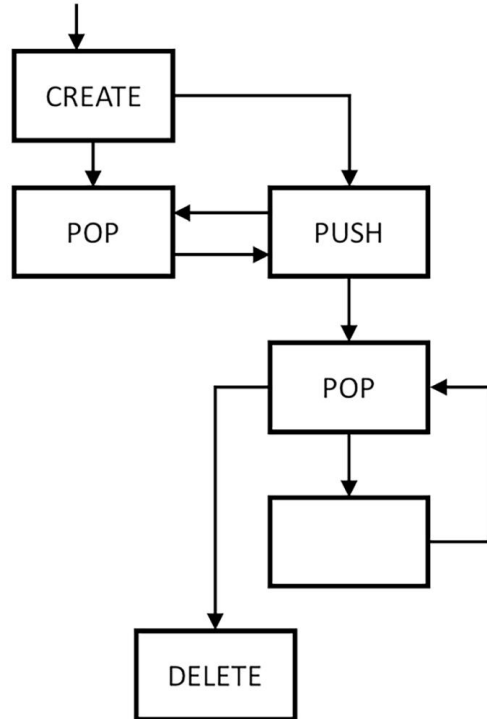
Ejemplo La Figura 3.8 muestra el diseño de una arquitectura de flujo de información en un Cierta estructura de datos utilizada en el sistema, llamada pila. Posibles operaciones en el pila son:

- CREATE: creación de una pila
- EMPUJAR: colocar un elemento encima de la pila
- POP: sacar un artículo de la parte superior de la pila
- ELIMINAR: elimina la estructura de pila del sistema.

El evaluador se encuentra actualmente en la fase de preparación individual. La lista de verificación utilizada por El probador consta de los siguientes elementos:

1. ¿Se realiza cada operación PUSH y POP solo después de que se haya creado la pila? (CREAR)?
2. ¿El sistema no permite la ejecución de POP en una pila vacía?

Fig. 3.8 Flujo del sistema de operaciones de pila



3. ¿El sistema prohíbe eliminar la pila (DELETE) cuando no está vacía?
4. ¿La cantidad de elementos de la pila nunca excederá los 10?

El evaluador utiliza una técnica de revisión basada en listas de verificación y verifica qué propiedades en la lista están satisfechos y cuáles no.

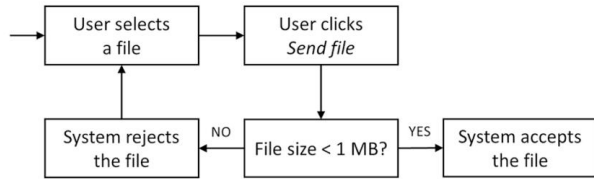
Se cumple el punto 1: cada instrucción PUSH y POP se crea después de la operación CREATE y no se puede ejecutar ninguna instrucción PUSH o POP después de la operación DELETE. Un evaluador curioso notará, sin embargo, que la especificación no dice si la pila está vacía cuando se crea o si contiene, por ejemplo, algunos elementos iniciales predeterminados. Vale la pena señalar al autor del documento que debe dejar esto claro en la especificación.

El punto 2 no se cumple. Después de la operación CREATE, la operación POP (en una pila vacía) se puede realizar inmediatamente.

Se cumple el punto 3. Después de cada operación PUSH, la siguiente operación debe ser una operación POP. Por lo tanto, solo se puede almacenar un elemento en la pila, que se eliminará de la pila en el siguiente paso. En este punto, el evaluador puede señalar que, dado que puede haber como máximo un elemento en la pila, ¿por qué usar una estructura de pila complicada cuando se podría usar una estructura más simple (como una variable o un registro) que nos permita almacenar solo uno? ¿elemento?

Se cumple el punto 4; así se desprende del análisis del punto 3.

Fig. 3.9 Modelo de proceso de negocio



Como puede ver en el ejemplo anterior, el revisor no se limitó a "marcar" elementos individuales de la lista, sino que también notó algunas deficiencias en la especificación. Esta es una valiosa retroalimentación de revisión que puede servir para mejorar el producto del trabajo (especificación) de su autor.

Ejemplo Un arquitecto le ha pedido a un desarrollador que lleve a cabo una revisión informal por pares.

La revisión se refiere a un modelo de proceso creado por el arquitecto que implementa el siguiente requisito comercial: "Un usuario puede cargar cualquier archivo de menos de 1 GB a través de un formulario web. Si el tamaño del archivo es menor que este valor, el sistema lo acepta; de lo contrario, rechaza el archivo".

El modelo revisado se muestra en la Fig. 3.9.

El promotor recibió del arquitecto tanto un requisito como un diagrama que muestra el proceso. El desarrollador verifica que el diagrama corresponda al requisito y observa que hay un error tipográfico menor, pero muy significativo, al decidir si el sistema debe aceptar o rechazar el archivo: la condición dice "Tamaño de archivo < 1 MB" pero debería decir "Archivo tamaño < 1 GB". El desarrollador informa al arquitecto del error tipográfico observado y el arquitecto corrige el diseño.

3.2.5 Factores de éxito de las revisiones

La clave para una revisión exitosa es la selección cuidadosa del tipo de revisión y las técnicas de revisión utilizadas. Además, se deben tener en cuenta una serie de otros factores que pueden afectar el resultado. Los factores de éxito incluyen, entre otros, los siguientes factores:

- Cada revisión tiene objetivos claros definidos durante la planificación que pueden servir como criterios de salida mensurables.
- Los tipos de revisiones utilizados son conducentes a lograr los objetivos establecidos y son apropiados para el tipo y nivel de los productos de trabajo de software y para los participantes.
- Se utilizan varias técnicas de revisión (como la revisión basada en listas de verificación o basada en roles). Se utiliza para identificar eficazmente los defectos presentes en un producto de trabajo.
- Las listas de verificación utilizadas están actualizadas y abordan los principales riesgos.
- Los documentos grandes se escriben y revisan en lotes, de modo que los autores obtengan comentarios rápidos y frecuentes sobre los defectos (lo cual es parte del control de calidad).

- Los participantes tienen tiempo suficiente para prepararse para la revisión; las revisiones están programadas por adelantado.
- Los autores reciben comentarios de los revisores, para que puedan mejorar el producto de su trabajo y la calidad de su trabajo. • La dirección apoya el proceso de revisión (por ejemplo, designando tiempo suficiente para revisar las actividades en el cronograma del proyecto).
- Las revisiones se consideran una parte natural de la cultura organizacional para promover aprendizaje y mejora de productos y procesos.
- La revisión involucra a personas cuya participación es propicia para lograr sus objetivos; por ejemplo, personas con diferentes habilidades o puntos de vista que potencialmente utilizarán el documento en el curso de su trabajo. • Los evaluadores son reconocidos como participantes importantes en la revisión y el conocimiento que obtienen sobre el producto del trabajo les permite preparar pruebas más efectivas con anticipación.
- Los participantes dedican tiempo suficiente para participar en la revisión y muestran la debida atención a los detalles.
- Las revisiones se realizan en piezas pequeñas para que los revisores no pierdan la concentración durante la revisión individual y/o la reunión de revisión (si se lleva a cabo). • Los defectos detectados se reconocen, confirman y tratan objetivamente. • Las reuniones de revisión se moderan de la manera correcta para que los participantes no desperdicien tiempo en actividades innecesarias.
- La revisión se lleva a cabo en una atmósfera de confianza mutua y sus resultados no son utilizado para evaluar a los participantes.
- Los participantes evitan gestos y comportamientos que puedan indicar aburrimiento, irritación u hostilidad hacia otros participantes. • Se brindó la debida capacitación a los participantes, especialmente para los tipos más formales de revisiones (como inspecciones).
- Se ha creado un ambiente propicio para ampliar el conocimiento y mejorar los procesos. sido creado.

Ejemplo Se le pidió a un líder de revisión que organizara una revisión del código del componente X (en forma de recorrido) para un grupo de desarrolladores. El líder de revisión realizó la inspección e invitó a participar al autor del componente X y a un equipo de evaluadores.

Este escenario carecía de factores de éxito tanto organizativos como relacionados con las personas:

- Se pidió al líder de revisión que organizara un recorrido, pero organizó una inspección, que no es el mejor tipo de revisión para una revisión de código. Por lo tanto, no se cumplió el factor de éxito organizacional (elegir el tipo de revisión apropiado para lograr los objetivos dados y para adaptarse al tipo de producto de trabajo, los participantes de la revisión, las necesidades y el contexto del proyecto).
- Se pidió al líder de revisión que organizara una revisión de código para desarrolladores, pero, a excepción del autor, sólo invitó a evaluadores. Una revisión de código de este tipo será ineficaz. Por tanto, no se ha cumplido el factor de éxito de carácter personal (la revisión involucra a personas cuya participación favorece el logro de sus objetivos).

3.2.6 (*) Técnicas de revisión

El programa de estudios de Foundation Level en la versión anterior (3.1) describía cinco técnicas diferentes que un revisor puede utilizar como parte de una actividad de revisión individual, es decir, preparación individual para la revisión. Por supuesto, esta actividad puede realizarse como parte de todos los tipos de revisión, en particular los cuatro tipos de revisión descritos en la Sección. [3.2.3](#). El propósito de estas técnicas es detectar defectos en el producto de trabajo revisado.

Las cinco técnicas mencionadas son:

- Revisión ad hoc •

Revisión basada en listas de

verificación • Escenarios y

simulacros • Revisión

basada en roles • Lectura basada en perspectivas

Revisión ad hoc

Este es el enfoque tradicional para la detección de defectos por parte de los revisores. El proceso está completamente desestructurado. Cada revisor tiene la tarea de detectar tantos defectos de todos los tipos posibles como sea posible. La eficacia de dicha revisión depende en gran medida de las habilidades de los revisores individuales. Generalmente conduce a la detección de los mismos problemas (generalmente obvios o triviales) por parte de muchos revisores diferentes.

Durante una revisión ad hoc, los revisores reciben poca (o ninguna) orientación sobre cómo realizar la tarea. Los participantes suelen leer el producto del trabajo de forma secuencial, identificando y documentando los problemas encontrados sobre la marcha.

Revisión basada en listas de

verificación Una revisión basada en listas de verificación es una técnica más estructurada que una revisión ad hoc. Una buena práctica para este tipo de revisión es que diferentes revisores reciban listas de verificación diferentes. Esto aumentará la cobertura potencial del producto y detectará más defectos. También reduce el riesgo de que más personas detecten el mismo problema varias veces, lo que supone una pérdida de tiempo y recursos. La ventaja más importante de la técnica basada en listas de verificación es la cobertura sistemática de los tipos de defectos más comunes.

Con este enfoque existe el riesgo de que los revisores se limiten estrictamente a las cuestiones de la lista de verificación e ignoren otros problemas potenciales en el producto de trabajo que se está revisando. Por lo tanto, los revisores deben ser conscientes de que tienen más responsabilidad que simplemente seguir ciegamente los elementos de la lista de verificación.

Las listas de verificación deben seleccionarse según el tipo de producto del trabajo y el propósito del equipo. Por lo tanto, una lista de verificación para el uso de una revisión de requisitos será bastante diferente de, por ejemplo, una lista de verificación para el uso de pruebas de seguridad o pruebas de usabilidad de la interfaz. La lista de verificación puede incluso ser específica del método particular utilizado para producir el producto del trabajo. Por ejemplo, una lista de verificación utilizada para probar productos relacionados con la banca puede incluir regulaciones legales impuestas a los bancos, mientras que una utilizada en la industria automotriz puede, a su vez, basarse en la norma ISO 26262 [40].

Un problema común que surge al utilizar este enfoque es que las listas de verificación se vuelven cada vez más largas con el tiempo y, por lo tanto, menos prácticas de usar. un típico

La lista de verificación no debe contener más de aprox. 10 elementos y deben revisarse y actualizarse periódicamente. Las actualizaciones deben abordar especialmente aquellos problemas, fallos y defectos que nosotros mismos hemos detectado en nuestro producto. Según nuestra experiencia, una lista de verificación de este tipo será mucho mejor (en términos de eficiencia de detección de defectos) que, por ejemplo, una lista de verificación estándar que se encuentra en Internet.

Un enfoque común es utilizar el análisis de riesgos y ampliar la lista de verificación para incluir los riesgos identificados con el nivel de gravedad más alto. Esto permite comprobar directamente los mayores riesgos cuando se utiliza un enfoque de lista de verificación.

En la sección 1 se presenta un ejemplo de revisión basada en una lista de verificación. [3.2.4](#) (un ejemplo de pila).

Escenarios y simulacros

El enfoque basado en escenarios funciona bien cuando los requisitos, el diseño o las pruebas mismas están documentados en un formato de "escenario" apropiado, como los casos de uso. En este enfoque, los revisores realizan los llamados ensayos en seco del producto de trabajo, verificando que la funcionalidad del producto se describa correctamente y que las excepciones comunes causadas por el mal comportamiento del producto se manejen adecuadamente.

Existe el riesgo de que los espectadores sigan demasiado de cerca los escenarios definidos. En tal situación, pueden pasar fácilmente por alto algunos defectos, como la falta de funcionalidad en el producto.

Al igual que con el enfoque de lista de verificación, los escenarios se pueden ampliar para incluir aspectos que surjan del análisis de riesgos para garantizar que los escenarios más importantes y utilizados con más frecuencia se exploren más a fondo.

Revisión basada en roles

La revisión basada en roles es una técnica en la que los revisores evalúan un producto de trabajo desde la perspectiva de los roles particulares de las partes interesadas. Los roles típicos incluyen tipos específicos de usuarios finales (experimentados, inexpertos, ancianos, niños, etc.) o roles específicos en la organización (usuario, administrador, administrador del sistema, evaluador de rendimiento, etc.).

A menudo, los roles se modelan mediante la llamada persona. Una persona es un personaje ficticio pero concreto que simboliza un determinado tipo de usuario. Al hacer que esta persona sea concreta (especificando, por ejemplo, su género, edad e intereses), es más fácil para el equipo "entrar en" el tipo de usuario. En la figura 3.10 se muestra un ejemplo de persona.

Lectura basada en perspectiva

La técnica de lectura basada en perspectiva se describe en la literatura como uno de los métodos de revisión individual más eficaces [41]. Se basa en el hecho de que diferentes revisores adoptan diferentes perspectivas de las partes interesadas al revisar un producto de trabajo y revisan el producto desde ese ángulo. Al considerar múltiples vistas del producto, los revisores pueden descubrir una mayor cantidad de problemas potenciales. Al mismo tiempo, al asignar perspectivas específicas a los revisores, cada revisor puede revisar el producto a fondo y en detalle, y dado que cada uno corresponde a una perspectiva diferente, el riesgo de detectar defectos duplicados es bajo.

La lectura basada en perspectiva no se limita sólo a adoptar diferentes puntos de vista. También requiere que los revisores intenten utilizar el producto de trabajo bajo revisión para generar un primer prototipo del producto para ver si es posible hacerlo con base en la información.



Benjamin Thompson

Position: Mobile applications tester
Age: 28 yo
City: New York
Hobbies: movies, FPS games, travelling, new technologies
Character traits: vigorous, impatient, curious

Fig. 3.10 Ejemplo de persona

disponible en el producto de trabajo revisado. Por ejemplo, los evaluadores intentarán generar versiones preliminares de las pruebas de aceptación si realizan una revisión basada en la perspectiva de la especificación de requisitos para verificar que la especificación contenga toda la información necesaria. Además, la lectura basada en la perspectiva suele utilizar listas de verificación.

Las perspectivas típicas que podría adoptar un espectador son:

- Usuario
- Analista de negocios •
- Diseñador •
- Tester •
- Administrador de sistemas •
- Gerente de marketing y promociones • Ingeniero de soporte técnico

Si la técnica se limita al punto de vista del usuario final, también se puede utilizar una técnica de lectura basada en perspectiva, distinguiendo entre tipos de usuarios del sistema, como por ejemplo:

- Usuario final •
- Administrador del sistema •
- Ingeniero de soporte técnico

Un factor clave para el éxito de la lectura basada en perspectivas es tener en cuenta y equilibrar los puntos de vista de las distintas partes interesadas de una manera que sea adecuada al riesgo. Los puntos de vista que adoptemos deberían depender del contexto. Por ejemplo:

- Si el documento que se revisa son requisitos, las perspectivas típicas serán usuario, diseñador y probador.
- Si el sistema está relacionado con un área altamente regulada (por ejemplo, aviación, banca, sistemas médicos), el punto de vista del regulador definitivamente debe incluirse en la revisión.
- Si el sistema se va a utilizar durante un largo tiempo, se debe adoptar el punto de vista del equipo de mantenimiento del sistema.

Revisión basada en roles versus lectura basada en perspectivas La

lectura basada en perspectivas es similar a la técnica de revisión basada en roles discutida anteriormente. Los dos enfoques a veces se equiparan en la literatura, pero difieren. La diferencia tal vez pueda explicarse de la forma más sencilla como sigue:

- En una revisión basada en roles, los "tipos" de usuarios cambian, pero las tareas a realizar siguen siendo los mismos (por ejemplo, diferentes tipos de clientes bancarios).
- En la lectura basada en perspectivas, las "perspectivas" de las partes interesadas o, en el caso de los propios usuarios, los tipos de tareas a realizar (por ejemplo, usuario final, administrador, analista de negocios, evaluador) cambian.

Ejemplo El equipo utiliza una técnica de lectura basada en perspectiva. La persona que revisa el documento (la especificación de requisitos) adopta la perspectiva del evaluador y utiliza el siguiente procedimiento para leer la especificación.

PROCEDIMIENTO DE LECTURA. Para cada requisito, cree una prueba o un conjunto de pruebas para garantizar que la implementación cumpla con el requisito. Utilice un enfoque de prueba estándar y criterios y técnicas de prueba para crear el conjunto de pruebas. Al crear pruebas para cada requisito, responda las siguientes preguntas:

1. ¿Tiene suficiente información para identificar el elemento de prueba y los criterios de prueba?
¿Puede crear casos de prueba razonables para cada elemento según estos criterios?
2. ¿Existe otro requisito para el cual podría generar un caso de prueba similar, pero
¿Con un resultado esperado diferente?
3. ¿Está seguro de que la prueba que generó proporcionará los valores correctos en el momento correcto?
¿unidades?
4. ¿Existen otras interpretaciones de este requisito que un desarrollador podría adoptar, según cómo se define el requisito? ¿Afectaría sus pruebas?
5. ¿Es el requisito razonable y racional en términos de su conocimiento de la aplicación y de lo que se incluye en la descripción general del sistema?

Preguntas de muestra

Pregunta 3.1

(FL-3.1.1, K1)

Su organización acaba de preparar un documento oficial que describe cómo las revisiones debe realizarse en la organización.

¿Se puede revisar este documento?

- R. Sí, porque cualquier documento comprensible para humanos puede someterse a pruebas estáticas.
- B. No, porque tendríamos que aplicarnos las reglas descritas en este documento durante la revisión.
- C. No, porque el documento no es un producto del trabajo ni del proceso de prueba ni del el proceso de desarrollo.
- D. No, porque solo se pueden realizar revisiones de especificaciones y código fuente.

Elija una respuesta.

Pregunta 3.2

(FL-3.1.2, K2)

Mientras analizaba el código, el evaluador notó que la complejidad ciclomática de uno de los componentes del código era muy alta. El evaluador pasó esta información a los desarrolladores, quienes refactorizaron el código, haciéndolo más legible y comprobable. Este escenario muestra el beneficio de utilizar:

- A. Pruebas dinámicas.
- B. Pruebas estáticas.
- C. Gestión de pruebas.
- D. Técnica de prueba formal.

Elija una respuesta.

Pregunta 3.3

(FL-3.1.3, K2)

¿Cuál es la diferencia entre técnicas estáticas y dinámicas, según el propósito?
de estas técnicas?

- R. Las técnicas estáticas detectan fallas directamente, mientras que las técnicas dinámicas detectan directamente detectar defectos.
- B. Las técnicas estáticas se suelen utilizar al principio del SDLC, mientras que las técnicas dinámicas
Las técnicas se utilizan normalmente en fases posteriores del SDLC.
- C. No hay diferencia, ya que ambos tipos de técnicas tienen como objetivo detectar defectos lo antes posible.
como sea posible.
- D. Las técnicas estáticas generalmente requieren habilidades de programación, mientras que las técnicas dinámicas
normalmente no lo hago.

Elija una respuesta.

Pregunta 3.4

(FL-3.2.1, K1)

Considere las siguientes afirmaciones sobre la retroalimentación temprana.

- i. La retroalimentación temprana brinda a los desarrolladores más tiempo para producir nuevas funciones del sistema, ya que dedican menos tiempo a modificar las funciones planificadas en una iteración determinada.
- ii. La retroalimentación temprana permite a los equipos ágiles ofrecer primero las funciones con mayor valor comercial, ya que la atención del cliente permanece centrada en las funciones más importantes desde el punto de vista del cliente. III. La retroalimentación temprana reduce el costo general de las pruebas porque reduce la cantidad de tiempo que los evaluadores necesitan para probar el sistema. IV. La retroalimentación temprana aumenta la probabilidad de que el sistema producido se acerque a las expectativas de los clientes, ya que tienen la oportunidad de realizar cambios durante cada iteración.

¿Cuáles de estas afirmaciones son ciertas?

- R. (i) y (iv) son verdaderos; (ii) y (iii) son falsos.
- B. (ii) y (iii) son verdaderos; (i) y (iv) son falsos.
- C. (ii) y (iv) son verdaderos; (i) y (iii) son falsos.
- D. (i) y (iii) son verdaderos; (ii) y (iv) son falsos.

Elija una respuesta.

Pregunta 3.5

(FL-3.2.2, K2)

¿Cuál de las siguientes es parte de la fase de inicio de revisión?

- A. Seleccionar quién participará en la revisión.
- B. Recopilación de métricas.
- C. Responder a las preguntas de los participantes sobre el alcance, los objetivos, el proceso, las funciones y el trabajo.
productos.
- D. Identificar el alcance del trabajo, incluido el tipo de revisión y los documentos (o partes de los
mismos) que son objeto de la revisión y las características de calidad a evaluar.

Elija una respuesta.

Pregunta 3.6

(FL-3.2.3, K1)

¿Qué función es responsable del buen funcionamiento de la reunión de revisión?

- A. Líder de revisión.
- B. Moderador.
- C. Autor.
- D. Revisor.

Elija una respuesta.

Pregunta 3.7

(FL-3.2.4, K2)

Durante los últimos días, el equipo ha estado intentando encontrar la causa de un extraño fallo del sistema. Como no pudieron descubrir la causa, el equipo decidió "simular la computadora" ejecutando manualmente el código línea por línea para comprender qué estaba haciendo exactamente el programa y así descubrir la causa del extraño comportamiento del software.

¿Qué tipo de revisión será la más apropiada para esta actividad?

- A. Inspección.
- B. Revisión informal.
- C. Revisión técnica.
- D. Tutorial.

Elija una respuesta.

Pregunta 3.8

(FL-3.2.5, K1)

La inspección se realizará mejor si:

- R. Su objetivo principal se definirá como "evaluación de alternativas".
- B. El gerente asistirá a las reuniones de revisión.
- C. Los revisores estarán capacitados para realizar revisiones.
- D. No se recopilarán métricas durante el proceso de revisión.

Elija una respuesta.

Capítulo 4 Análisis y diseño de pruebas



Palabras clave

Criterios de aceptación	Los criterios que un componente o sistema debe satisfacer para ser aceptado por un usuario, cliente u otra entidad autorizada. Referencias: ISO 24765.
Desarrollo basado en pruebas de aceptación:	un enfoque de prueba primero basado en la colaboración que define las pruebas de aceptación en el lenguaje de dominio de las partes interesadas. Abreviatura: ATDD. una
Técnica de prueba de caja negra	técnica de prueba basada en un análisis de la especificación de un componente o sistema. Sinónimos: técnica de diseño de pruebas de caja negra, técnica basada en especificaciones. una
Análisis de valor límite	técnica de prueba de caja negra en la que los casos de prueba se diseñan en función de valores límite. la
Cobertura de sucursales	cobertura de sucursales en un gráfico de flujo de control.
Pruebas basadas en listas de verificación	una técnica de prueba basada en la experiencia en la que se diseñan casos de prueba para ejercitar los elementos de una lista de verificación.
Enfoque de prueba basado en la colaboración:	un enfoque de prueba que se centra en evitar defectos mediante la colaboración entre las partes interesadas. el
Cobertura	grado en que los elementos de cobertura especificados son ejercidos por un conjunto de pruebas, expresado como porcentaje. Sinónimos: cobertura de prueba.

Artículo de cobertura	<p>un atributo o combinación de atributos derivado de una o más condiciones de prueba</p> <p>mediante el uso de una técnica de prueba.</p>
Prueba de tabla de decisiones	<p>una técnica de prueba de caja negra en la que se prueba</p> <p>Los casos están diseñados para ejercer la combinaciones de condiciones y acciones resultantes mostradas en una decisión mesa.</p>
Partición de equivalencia	<p>una técnica de prueba de caja negra en la que se prueba las condiciones son particiones de equivalencia ejercido por un miembro representativo de cada partición. Según ISO 29119-1.</p> <p>Sinónimos: prueba de partición.</p>
Error al adivinar	<p>una técnica de prueba en la que se derivan pruebas sobre la base del conocimiento del evaluador sobre fracasos pasados o conocimiento general de modos de fallo. Referencias: ISO 29119-1.</p>
Técnica de prueba basada en la experiencia	<p>una técnica de prueba basada en la experiencia del probador. experiencia, conocimiento e intuición.</p> <p>Sinónimos: diseño de pruebas basado en la experiencia técnica, técnica basada en la experiencia.</p>
Prueba exploratoria	<p>un enfoque de prueba en el que los evaluadores diseñar y ejecutar pruebas dinámicamente basándose en su conocimiento, exploración de el elemento de prueba y los resultados de pruebas anteriores. pruebas. Según ISO 29119-1.</p>
Pruebas de transición estatal	<p>una técnica de prueba de caja negra en la que se prueba</p> <p>Los casos están diseñados para ejercitar elementos de Un modelo de transición estatal. Sinónimos: finito pruebas estatales.</p>
Cobertura de declaración	<p>la cobertura de declaraciones ejecutables.</p>
Técnica de prueba	<p>un procedimiento utilizado para definir las condiciones de prueba, diseñar casos de prueba y especificar datos de prueba.</p> <p>Sinónimos: técnica de diseño de pruebas.</p>
Técnica de prueba de caja blanca	<p>una técnica de prueba basada únicamente en el interior estructura de un componente o sistema.</p> <p>Sinónimos: diseño de prueba de caja blanca técnica, técnica basada en estructuras.</p>

4.1 Descripción general de las técnicas de prueba


FL-4.1.1 (K2) Distinguir técnicas de prueba de caja negra, de caja blanca y basadas en la experiencia.

En esta sección, describimos:

- Cómo elegir la técnica de prueba adecuada •
- Qué tipos de técnicas de prueba se describen en el programa de estudios del nivel básico •
- ¿Cuáles son las características comunes de cada grupo de técnicas de prueba?

Según la definición del diccionario Merriam-Webster, una técnica (del gr. *techné* (τέχνη)—arte, artesanía, maestría, habilidad) es "un conjunto de métodos técnicos (como en un oficio o en una investigación científica)" o "un método para lograr un **objetivo** deseado".

En el contexto de las pruebas de software, esta actividad consistirá en utilizar ciertos métodos para derivar (producir) condiciones de prueba, casos de prueba y datos de prueba basados en la base científica del conocimiento del software.

Digamos de inmediato que cada técnica de prueba.  está estrictamente definido, es decir, es formal en el sentido de los principios que lo constituyen (aunque en sí mismo, como tal, formal no necesita serlo, por ejemplo, conjeturas erróneas o pruebas exploratorias). Esto se debe a que detrás de cada técnica de prueba hay lo que Glenford Myers llama una "hipótesis de error". [10]. En el contexto del vocabulario ISTQB® [42], tal vez un nombre más apropiado sería "hipótesis del defecto", pero por razones históricas, nos quedaremos con el nombre original (de todos modos, las técnicas de prueba también detectan defectos debidos a errores de programación específicos).).

Esta hipótesis del error, o, como a veces se la llama, teoría del error, es la base científica para la construcción de cada técnica. Al analizar cada uno de ellos en las siguientes subsecciones, prestaremos especial atención a qué tipos de problemas es capaz de detectar la técnica. Por lo tanto, las técnicas de prueba son independientes de las tecnologías específicas, las herramientas utilizadas o el tipo de software bajo prueba. Cada técnica se basa en un problema abstracto relacionado con una hipótesis de error específica y está diseñada para detectar ese tipo específico de error o defecto.

El programa de estudios de Foundation Level analiza nueve técnicas de prueba y las agrupa en tres categorías:

- Técnicas de prueba de caja negra (cuatro técnicas)
- Técnicas de prueba de caja blanca (dos técnicas) •
- Técnicas de prueba basadas en la experiencia (tres técnicas)

Esta división se realiza en función de la fuente de conocimiento sobre el objeto de prueba. La figura 4.1 justifica esquemáticamente la razón de tal división de técnicas.

Antes de desarrollar un software, hay que diseñarlo. La función de la documentación de diseño puede cumplirse mediante especificaciones de requisitos, casos de uso o procesos comerciales.

¹<https://www.merriam-webster.com/dictionary/technique>

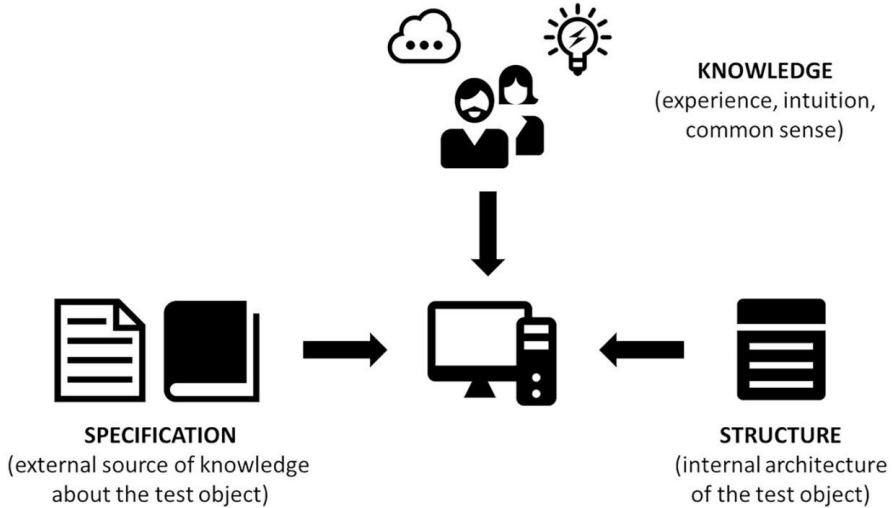


Fig. 4.1 Fuentes de conocimiento sobre el objeto de prueba como base para la categorización de técnicas de prueba

descripción. A partir de ellos se crea un software que tiene una estructura específica. Es, por ejemplo, código, estructura de menú, estructura de flujo de procesos de negocio u otra forma de descripción de la arquitectura. Además de estas fuentes formales de conocimiento, también existen otras menos formales, relacionadas directamente con las personas que trabajan en el desarrollo de software. Estas personas tienen conocimientos, experiencia e intuición, de las que también se pueden derivar pruebas valiosas.

Ahora discutiremos las diferentes categorías de técnicas de prueba con más detalle.

Técnicas de prueba de caja negra

Las técnicas de prueba de caja negra también se conocen como técnicas de comportamiento o técnicas basadas en especificaciones. Utilizan conocimientos externos al objeto de prueba sobre cómo debería comportarse. Este conocimiento puede ser, por ejemplo, especificación de requisitos, descripción de casos de uso, historias de usuarios (en proyectos ágiles) y procesos de negocio. Todos estos modelos describen el comportamiento deseado del sistema, tanto funcional como no funcional, sin hacer referencia a su diseño interno.

La ventaja de utilizar técnicas de caja negra es que los documentos antes mencionados generalmente existen mucho antes de que comience la implementación de un componente o sistema. Esto significa que las actividades de prueba (por ejemplo, análisis y diseño de pruebas) pueden comenzar mucho antes del desarrollo del código. Esto nos permite utilizar técnicas de prueba de caja negra no solo durante las pruebas dinámicas sino también en la etapa de diseño de las pruebas, como una especie de método de prueba estático. Las técnicas de caja negra se pueden utilizar tanto en pruebas funcionales como no funcionales.

Técnicas de prueba de caja blanca

Las técnicas de prueba de caja blanca también se denominan técnicas estructuradas o basadas en estructura. La base del diseño de pruebas de caja blanca es la estructura interna del objeto de prueba.

Muy a menudo, esta estructura es el código fuente, pero también puede ser otro modelo de más alto nivel del sistema o arquitectura del módulo. Por ejemplo, en el nivel de prueba de integración, dicho modelo puede ser el llamado gráfico de llamadas, y en el nivel de prueba del sistema, el flujo de información en el proceso de negocio.

Un lector atento puede notar que dado que, cuando probamos un programa, nos referimos al programa mismo (por ejemplo, al código fuente), llegamos a una situación paradójica en la que el programa se convierte en su propio oráculo, es decir, decide por sí mismo cómo debería comportarse. De hecho, este es un problema aparente. El conocimiento de la estructura interna del programa se utiliza sólo para diseñar pruebas que cubran elementos específicos de este modelo (por ejemplo, declaraciones de código, decisiones en el código, rutas en el programa). Por el contrario, para cada prueba, su resultado esperado debe determinarse sobre la base de conocimientos externos al sistema bajo prueba, por ejemplo, sobre la base de especificaciones o sentido común.

El código nunca puede ser un oráculo en sí mismo.

Técnicas de prueba basadas en la experiencia El

grupo de técnicas de prueba basadas en la experiencia se diferencia de los otros dos grupos en que no se basa en ningún documento formal: diseño, requisitos, código, etc. Esto se debe a que las técnicas de prueba basadas en la experiencia utilizan algo más "fuentes blandas de conocimiento sobre el sistema bajo prueba. Estas fuentes son conocimiento, intuición, experiencia, conocimiento de defectos encontrados en versiones anteriores del sistema o aplicaciones similares, etc. Por lo tanto, se refieren directamente a las cualidades y habilidades de los propios evaluadores, en lugar de conocimiento "objetivo" sobre el sistema bajo prueba.

Las técnicas pueden aprovechar no sólo la experiencia de los evaluadores sino también de todas las demás partes interesadas del proyecto. Los ejemplos incluyen el uso de la experiencia de desarrolladores, usuarios finales, clientes, arquitectos, analistas de negocios y gerentes de proyectos. Las técnicas de prueba basadas en la experiencia se utilizan a menudo en combinación con técnicas de prueba de caja negra y caja blanca. Esto brinda a los evaluadores la oportunidad de detectar problemas que fácilmente se pasan por alto mediante el uso de técnicas de prueba más formales, que no pueden tener en cuenta todos los matices del proyecto o el contexto en el que se desarrolla el software.

La Tabla 4.1 muestra las diferencias básicas entre los tres grupos de técnicas de prueba mencionado anteriormente.

Como se mencionó anteriormente, el programa de estudios de Foundation Level introduce nueve técnicas de prueba, cuyo conocimiento es obligatorio para el examen. Éstas incluyen cuatro técnicas de caja negra, dos técnicas de caja blanca y tres técnicas basadas en la experiencia. En la figura 4.2 se muestra un resumen de estas técnicas. El programa de estudios no describe estas técnicas en detalle ni da ejemplos de su uso práctico, aunque la capacidad de aplicarlas correctamente es una de las cualidades más importantes de un buen evaluador. Por lo tanto, en las siguientes subsecciones las discutiremos con gran detalle, destacando los aspectos importantes de cada técnica. Para hacer el material más accesible, ilustraremos el uso de estas técnicas con muchos ejemplos.

Por supuesto, existen muchas otras técnicas de prueba. El programa de estudios analiza sólo los más populares y utilizados. Las técnicas de prueba y las medidas de cobertura correspondientes se describen en la norma internacional ISO/IEC/IEEE 29119-4 [4]. También se proporciona más información sobre técnicas de prueba en [16, 43–48].



Tabla 4.1 Comparación de categorías de técnicas de prueba

Criterio de comparación	Caja negra	Caja blanca	Basado en la experiencia
Fuente para derivar condiciones de prueba, casos de prueba y datos de prueba.	Bases de prueba externas al objeto de prueba: requisitos, especificaciones, casos de uso, historias de usuarios	Base de prueba que describe la estructura interna del objeto de prueba: código, arquitectura, diseño detallado; Las especificaciones se utilizan a menudo para describir los resultados esperados.	Conocimiento, experiencia, intuición de los evaluadores, desarrolladores, usuarios y otras partes interesadas.
Tipo de problemas detectados	Discrepancias entre los requisitos (comportamiento declarado) y su implementación.	Problemas asociados con el flujo de control o el flujo de datos.	Depende de la persona que realiza las pruebas o, por ejemplo, de la lista de verificación utilizada en el pruebas
Cobertura	Medido contra los elementos probados de la base de prueba y la técnica aplicada a la base de prueba.	Medido contra elementos probados de la estructura, como código o interfaces.	No definida

TEST TECHNIQUES		
BLACK-BOX	WHITE-BOX	EXPERIENCE-BASED
equivalence partitioning (EP)	statement testing and statement coverage	error guessing
boundary value analysis (BVA)	branch testing and branch coverage	exploratory testing
decision table testing		checklist-based testing
state transition testing		

Fig. 4.2 Técnicas de evaluación descritas en el programa de estudios del nivel básico

Selección de la técnica de prueba

Muchos factores influyen en la elección de las técnicas de prueba. Estos se pueden dividir en tres grupos principales:

- Factores formales (por ejemplo, documentación, leyes y regulaciones vigentes, disposiciones contractuales con el cliente, procesos implementados en la organización, objetivos de prueba, modelo SDLC utilizado).

- Factores del producto (por ejemplo, software, su complejidad, importancia de diversas características de calidad, riesgos, tipos esperados de defectos, uso esperado del software).
- Factores del proyecto (por ejemplo, tiempo disponible, presupuesto, recursos, herramientas, habilidades, conocimientos) y experiencia de los probadores).

Ejemplo Está trabajando en un proyecto que implica el desarrollo de una aplicación de biblioteca universitaria. En su organización, existe la obligación de probar las aplicaciones que se están desarrollando, mientras que la estrategia de prueba (ver Sección 5.1.1) implementada en su proyecto impone, debido a los contratos firmados con el cliente, la necesidad de realizar pruebas utilizando pruebas formales. técnicas para que el diseño, implementación, ejecución y resultados de las pruebas puedan documentarse. El proyecto se lleva a cabo bajo el modelo V.

Los requisitos se han recopilado y presentado en forma de especificación de requisitos. Actualmente, los arquitectos del sistema están trabajando en el diseño de la lógica de negocios del módulo responsable de verificar cuántos libros puede pedir prestado un usuario según este tipo de usuario (estudiante, empleado) y multas vencidas. Este es un componente clave para garantizar la adecuada implementación de las reglas y regulaciones de la biblioteca, por lo que el impacto del riesgo de su mal funcionamiento es muy alto.

Tienes 3 días para analizar y diseñar pruebas manuales. No se espera la automatización de pruebas debido a que las pruebas de regresión no desempeñarán un papel importante en este proyecto. Además, la organización no cuenta con herramientas especializadas ni las habilidades técnicas para crear scripts de prueba automatizados.

De la descripción anterior se desprende claramente que en el contexto del componente responsable de aplicar las normas de préstamo de libros:

- Necesitamos realizar pruebas (esto se desprende directamente de la documentación y contrato).
- Necesitamos diseñar y documentar estas pruebas (ídem).
- Las pruebas deben centrarse en la lógica empresarial, por lo que una opción sensata sería utilizar técnicas de prueba apropiadas para este aspecto del software.
- El impacto del riesgo asociado con este componente es alto, por lo que tiene sentido utilizar métodos que verifiquen la lógica empresarial con mucho cuidado.
- Las pruebas se realizarán manualmente (falta de herramientas y experiencia, bajo papel de las pruebas de regresión, que suelen ser un candidato natural para la automatización).

Como puede verse en el ejemplo anterior, la decisión sobre la elección de la técnica, el nivel de detalle del análisis, la elección del diseño, implementación y ejecución de la prueba está influenciada por muchos factores formales, de producto y de proyecto.

No existe una técnica de prueba perfecta y universal. Cada técnica tiene sus propias ventajas y desventajas. Cada uno se desempeñará mejor en una situación y peor en otra. Por ejemplo, en el caso de la situación antes mencionada con un componente

Tabla 4.2 Ejemplos de técnicas de prueba con diferentes niveles de formalización

Grado de formalización	Ejemplo
Muy bajo	Ejecución no planificada e indocumentada de adivinación de errores, sin guardar los resultados de las pruebas.
Bajo	Realización de pruebas exploratorias con una carta de prueba.
Medio	Utilizar pruebas de tablas de decisiones para probar la lógica empresarial; Los casos de prueba se documentan en las especificaciones de casos de prueba y los resultados de las pruebas se registran.
Grande	Utilizar el modelo de máquina de estados y la técnica de prueba de transición de estados para probar el comportamiento del programa; El diseño de la prueba (máquina de estado), los casos de prueba (en forma de scripts de prueba) y los resultados de la prueba (registros) están documentados.

que verifica las reglas para el préstamo de libros en una biblioteca, una de las técnicas de prueba de caja negra (prueba de tabla de decisiones (ver Sección 4.2.3)) parece ser una buena opción, porque prueba la lógica de negocios y esta es la funcionalidad. Lo que más nos importa son las pruebas. Por otro lado, no tiene sentido utilizar, por ejemplo, técnicas de prueba de caja blanca que se basan en la estructura interna del programa (código fuente), ya que el problema de prueba aquí es la prueba de lógica de negocios (las técnicas antes mencionadas se analizan en detalle más adelante en este capítulo).

Una buena práctica, que suelen utilizar los evaluadores experimentados, es combinar técnicas. Por ejemplo, al utilizar tablas de decisión en el problema anterior, el evaluador puede en ocasiones aplicar la técnica de análisis de valores límite (consulte la Sección 4.2.2) y verificar las reglas comerciales para dichos valores (por ejemplo, el número máximo posible de libros que un estudiante puede pedir prestados).

El uso de técnicas de prueba también puede considerarse en el contexto de los niveles de prueba. Algunas técnicas son más universales; por lo tanto, están naturalmente presentes en todos los niveles de prueba, desde las pruebas de componentes hasta las pruebas de aceptación. Algunas otras técnicas, por otra parte, tienen un ámbito de aplicación algo más limitado; por ejemplo, las técnicas de caja blanca se aplican con mayor frecuencia en el nivel de prueba de componentes (por ejemplo, pruebas unitarias de desarrollador). Por supuesto, se pueden imaginar ejemplos del uso de esta técnica a nivel de prueba de aceptación, pero en la práctica estas situaciones son bastante raras.

El grado de formalización del desarrollo de pruebas utilizando técnicas de prueba puede variar desde muy informal hasta muy formal. En el Cuadro 4.2 damos algunos ejemplos de los distintos grados de formalización.


4.2 Técnicas de prueba de caja negra

- FL-4.2.1 (K3) Utilice la partición de equivalencia para derivar casos de prueba.
- FL-4.2.2 (K3) Utilice el análisis de valores límite para derivar casos de prueba.
- FL-4.2.3 (K3) Utilice pruebas de tablas de decisión para derivar casos de prueba.
- FL-4.2.4 (K3) Utilice pruebas de transición de estado para derivar casos de prueba.

En esta sección, además de analizar las cuatro técnicas de prueba de caja negra descritas en el programa de estudios, analizamos una quinta técnica adicional: las pruebas basadas en casos de uso (Sección 4.2.5). Se trata de una materia optativa y no examinable. Esta técnica estaba presente en una versión anterior del programa de estudios. Creemos que es tan importante y tan utilizado que vale la pena discutirlo, incluso como contenido superobligatorio en relación al plan de estudios.

4.2.1 Partición de equivalencia (EP)

Uno de los siete principios de prueba presentados en la Sección. 1.3 dice que "las pruebas exhaustivas son imposibles". Esto es bastante obvio: el número de combinaciones posibles de datos de entrada es prácticamente infinito, mientras que el evaluador tiene la capacidad de realizar sólo un número finito y muy pequeño de pruebas.

La partición de equivalencia  La técnica intenta superar el principio de imposibilidad de realizar pruebas exhaustivas. Generalmente hay un número infinito de entradas posibles, pero el número de comportamientos esperados de un programa en estas entradas suele ser finito, especialmente si consideramos comportamientos específicos relacionados con un aspecto estrictamente definido del funcionamiento de la aplicación. Veamos algunos ejemplos.

Ejemplo El sistema establece el monto del impuesto según los ingresos. El impuesto puede ser del 0%, 19%, 33% o 45%. Hay potencialmente infinitos valores posibles para el ingreso, pero sólo cuatro tipos posibles de decisiones tomadas por el sistema.

Ejemplo Un usuario completa un formulario web y luego quiere imprimirlo. La impresión puede ser por una o ambas caras. Hay formas potencialmente infinitas de completar un formulario, especialmente si es complejo. Sin embargo, lo que queremos probar en este caso son sólo dos tipos de comportamiento: impresión correcta a una cara e impresión correcta a dos caras.

Ejemplo El usuario completa el formulario, incluido el campo "edad". A continuación, el usuario podrá comprar un billete, siendo el procedimiento diferente en función de la edad del usuario. El diagrama de flujo exacto del proceso se muestra en la Fig. 4.3. Hay muchas posibilidades para rellenar el formulario, pero sólo cuatro acciones posibles: mensaje de error, comprar entrada por un adulto, comprar entrada por un menor y cerrar sesión.

Como puede ver, normalmente es posible reducir el comportamiento de prueba de un programa a un número finito de variantes. El método de partición de equivalencia divide un dominio determinado en subconjuntos llamados particiones (o particiones de equivalencia) de manera que por cada dos elementos de una partición, tenemos un comportamiento de programa idéntico. Por ejemplo, si el sistema asigna un descuento a personas menores de 18 años, todos los números menores de 18 años formarán una partición de equivalencia, correspondiente a la asignación del descuento. De esta forma, desde el punto de vista del evaluador, los valores que pertenecen a la misma partición se tratan de la misma manera. Por lo tanto, cada elemento de una partición determinada es una opción igualmente buena para probar.

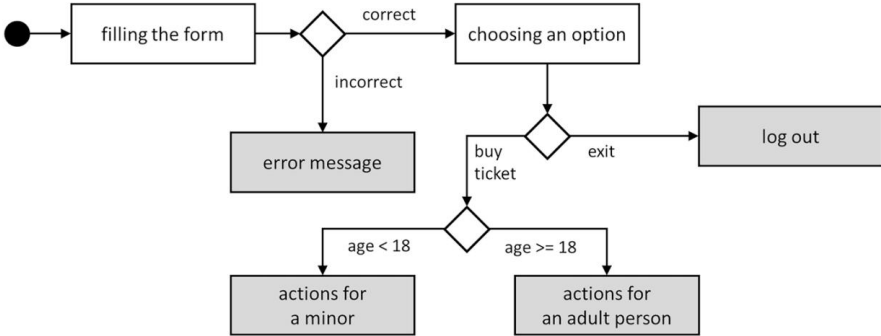


Fig. 4.3 Ejemplo de un proceso que describe posibles escenarios de uso del sistema

Aplicación La

técnica EP es versátil. Se puede utilizar prácticamente en cualquier situación, en cualquier nivel de prueba y en cualquier tipo de prueba. Esto se debe a que todo se reduce a la división de posibles datos en grupos. Vale la pena mencionar que la técnica se puede aplicar no sólo a dominios de entrada sino también a dominios de salida y dominios internos (es decir, aquellos relacionados con variables que no se dan directamente en la entrada ni se devuelven en la salida).

El dominio que dividimos en particiones de equivalencia no tiene por qué ser un dominio numérico. De hecho, puede ser cualquier conjunto no vacío. A continuación se muestran algunos ejemplos de dominios a los que se puede aplicar la técnica de partición de equivalencia:

- Un conjunto de números naturales (p. ej., partición en números pares e impares) •
- Una colección de palabras (p. ej., partición por longitud de palabra, una letra, dos letras, etc.) •
- Colecciones relacionadas con el tiempo (p. ej., partición por año de nacimiento, por mes en un determinado año, etc)
- Una colección de tipos de sistemas operativos: {Windows, Linux, macOS} (por ejemplo, partición en particiones de un solo elemento, {Windows}, {Linux}, {macOS})

Corrección de partición

Es muy importante que el particionamiento que hagamos sea correcto, lo que significa que:

- Cada elemento del dominio pertenece exactamente a una partición de equivalencia. •
- Ninguna partición de equivalencia está vacía.

La Figura 4.4 ilustra la cuestión de la corrección de la partición. Los puntos negros indican los elementos del dominio y los rectángulos grises indican las particiones de equivalencia. La Figura 4.4a muestra la partición de equivalencia correcta, que cumple las dos condiciones anteriores. La Figura 4.4b muestra una partición incorrecta: un elemento pertenece a dos particiones de equivalencia, lo que viola las condiciones de corrección.

La Figura 4.4c muestra otra partición incorrecta: uno de los elementos del dominio no ha sido asignado a ninguna partición de equivalencia.

La cuestión de la corrección de la partición parece bastante obvia, pero en la práctica no es un problema trivial. En aplicaciones reales, los dominios y las particiones pueden ser muy complejos.

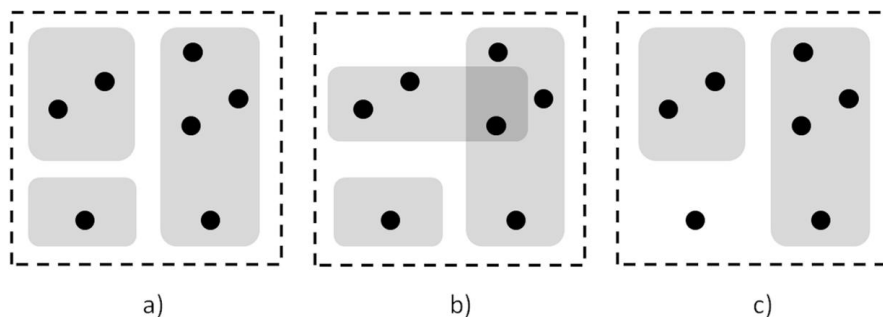


Fig. 4.4 Ejemplos de partición de dominio correcta (a) e incorrecta (b, c)

y tienen una estructura muy complicada. A menudo sucede que para una partición que hemos realizado, se violan algunas condiciones de corrección.

Ejemplo Consideremos el problema de prueba clásico descrito por Myers [10]. El programa toma tres números enteros no negativos, a , b , c , como entrada y genera el tipo de triángulo que se puede construir a partir de segmentos de las longitudes dadas. Las posibles respuestas del programa son "triángulo equilátero", "triángulo isósceles", "triángulo escaleno" y "no es un triángulo". Supongamos que queremos aplicar la técnica EP al dominio de entrada (es decir, al conjunto de todos los triples posibles (a, b, c) de enteros no negativos), respecto al dominio de salida (es decir, el tipo de triángulo). Parece natural dividir el dominio de entrada en las cuatro particiones correspondientes a las cuatro posibles salidas mencionadas anteriormente. Sin embargo, tras una inspección más cercana, resulta que todo triángulo equilátero es también un triángulo isósceles, por lo que estamos tratando con una partición de equivalencia, que es un subconjunto propio de otra partición. Necesitamos corregir nuestra partición, definiendo las siguientes particiones:

- Entradas que representan triángulos equiláteros
- Entradas que representan triángulos isósceles, que no son equiláteros •
- Entradas que representan triángulos
- escalenos • Entradas que caen en la categoría "no es un triángulo"

Ahora bien, esta partición es correcta: cada triple de números dados a la entrada corresponde exactamente a una de las cuatro posibles particiones anteriores.

Ejemplo Queremos clasificar el conjunto de todas las posibles secuencias finitas de números en función de su orden. Una partición de equivalencia natural de dichos datos en particiones de equivalencia podría verse así: secuencias ascendentes, secuencias descendentes y secuencias desordenadas. Pero tenga en cuenta que una secuencia de un elemento es tanto ascendente como descendente. Además, surge la pregunta de dónde clasificar la cadena vacía (que contiene 0 elementos). Por lo tanto, la partición correcta debe tener en cuenta estos "casos extremos" y las particiones pueden verse así:

- Partición 1: la secuencia vacía • Partición
- 2: todas las secuencias de un elemento • Partición
- 3: todas las secuencias ascendentes con al menos dos elementos • Partición 4:
- todas las secuencias descendentes con al menos dos elementos • Partición 5: todas
- las secuencias desordenadas con más de dos elementos

Las particiones que contienen valores "normales" y "correctos", es decir, valores esperados (aceptados) por el sistema, se denominan particiones válidas. Las particiones que contienen valores que el componente o sistema debería rechazar (por ejemplo, datos con sintaxis incorrecta, que exceden los rangos aceptables, etc.) se denominan particiones no válidas.

En nuestro último ejemplo, las cinco particiones que hemos identificado son válidas porque cada una de ellas contiene los datos correctos que espera el sistema (quizás el caso de la cadena vacía sea controvertido; si la especificación no menciona explícitamente cadenas no vacías, entonces la partición 1 puede considerarse una partición no válida). Además de estas particiones identificadas, podemos distinguir, por ejemplo, una partición no válida formada por elementos que no son cadenas numéricas (por ejemplo, contienen caracteres alfanuméricos).

Las definiciones de particiones válidas e inválidas, y en consecuencia de valores válidos e inválidos, pueden entenderse de diferentes maneras, por lo que si vamos a utilizar estos términos, vale la pena definirlos con precisión. Por ejemplo, los valores válidos se pueden entender al menos de dos maneras:

- Como aquellos que deben ser procesados por el sistema. •
- Como aquellos para los cuales la especificación define su procesamiento.

Del mismo modo, se pueden entender valores incorrectos:

- Como aquellos que deben ser ignorados o rechazados por el sistema. •
- Como aquellos para los cuales la especificación no define su procesamiento.

Ejemplo En el formulario de registro de usuario, el sistema espera que se ingrese una dirección de correo electrónico válida en el campo "correo electrónico". Cuando el usuario ingresa allí la cadena de caracteres "abc@def@ghi", el sistema rechaza esta entrada como inválida, informándole al usuario con el mensaje "Correo electrónico incorrecto". En esta situación, la entrada "abc@def@ghi" puede ser tratada por algunos como un valor no válido (debido a que el sistema lo rechazó, el registro se realizará solo si se proporciona la dirección de correo electrónico correcta) y por otros como un valor. proveniente de la partición válida (debido a que el sistema está preparado para este tipo de situaciones, la prueba de ello es el mensaje de error; por lo que es en cierto sentido un valor "esperado" y, por lo tanto, proveniente de la partición válida).

Mayor división de particiones en subparticiones Una ventaja importante de la técnica EP es que si divide alguna (incluso todas) particiones en subparticiones, aún tendrá una partición equivalente. Las pruebas para un dominio tan fragmentado serán más precisas. El evaluador puede decidir que ciertas particiones deben subdividirse aún más por determinadas razones. Este es un enfoque natural que aprovecha la naturaleza jerárquica de las particiones. Considere el siguiente ejemplo.

Ejemplo PESEL es el número de identificación personal de ciudadano utilizado por el gobierno polaco. Cada ciudadano polaco tiene asignado un número PESEL único. Es un número de 11 dígitos, en los que los primeros seis dígitos codifican la fecha de nacimiento, el último es el dígito de control y la imparidad del penúltimo dígito codifica el género: masculino o femenino.² El sistema toma el número PESEL del usuario, y, dependiendo de si la persona es mayor de edad o no, toma las medidas oportunas. Queremos realizar una partición de equivalencia para todas las secuencias posibles de dígitos. Como primer paso, podemos dividirlos en números válidos y no válidos, es decir, números que representan números PESEL válidos y todo lo demás. A continuación, podemos considerar cada una de estas particiones en términos de una posible división adicional. Sin duda, los números PESEL correctos deben dividirse, según la especificación, en números correspondientes a adultos y menores. Una división adicional de cada una de estas particiones podría, por ejemplo, tener en cuenta la codificación adecuada del número del mes (por ejemplo, para los nacidos entre 1900 y 1999, el mes se codifica normalmente; para los nacidos entre 2000 y 2099, 20 es agregado al número del mes (por ejemplo, febrero se codifica como 22, para 2100 a 2199, se agrega 40; para 2200 a 2299, se agrega 60 y para 1800 a 1899, se agrega 80);

A su vez, el número PESEL puede ser incorrecto por diversas razones, como estructurales (por ejemplo, longitud incorrecta) o sustantivas (por ejemplo, inconsistencia en el número de cheque). En la Fig. 4.5 se muestra un ejemplo del proceso de división de particiones en subparticiones.

Tengamos en cuenta que las personas nacidas entre 2000 y 2099 pueden ser tanto adultos como menores (suponiendo que hagamos la división en 2024), por lo que dividimos este período de tiempo en 2000-2006 y 2007-2099. Asumimos, a efectos de prueba, que las personas nacidas después de 2024 (lo que obviamente es imposible en 2024) se clasifican como menores. En las pruebas, sólo nos preocuparemos de verificar si se aceptan los números PESEL con la codificación de mes correcta, y queremos verificar esto para todas las codificaciones definidas en la especificación PESEL (1800-1899, 1900-1999, 2000-2099, 2100-2199, 2200-2299). Finalmente, como resultado de la división jerárquica de particiones, dividimos el dominio relacionado con los números PESEL en 12 particiones de equivalencia:

- Tres particiones válidas para adultos (1800-1899, 1900-1999, 2000-2006)
- Tres particiones válidas para menores (2007-2099, 2100-2199, 2200-2299)
- Tres particiones no válidas debido a la sintaxis (estructura) de el número PESEL
- Tres particiones no válidas debido a la semántica (significado) del número PESEL

Note que podríamos ir más lejos con la división. Por ejemplo, cada partición de equivalencia válida podría dividirse en dos particiones, codificando los géneros "masculino" y "femenino".

Derivación de casos de

prueba En el caso de la técnica EP, los elementos de cobertura son las particiones de equivalencia.

El conjunto mínimo de casos de prueba para garantizar una cobertura del 100% es uno que cubra cada partición de equivalencia, es decir, para cada partición identificada, hay un caso de prueba que contiene

²El sistema PESEL es bastante antiguo; se introdujo en la década de 1970 y, por lo tanto, no tenía en cuenta otros géneros que el masculino y el femenino.

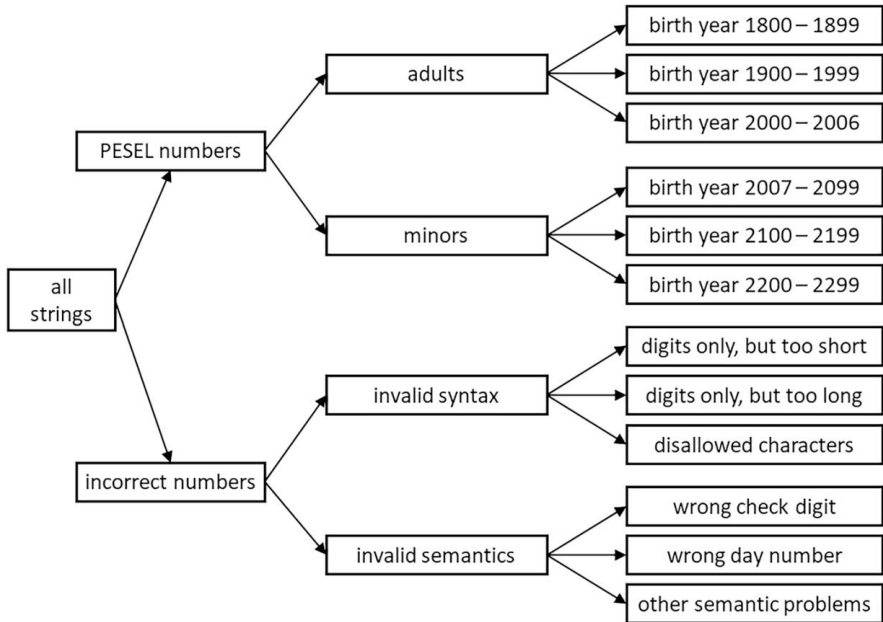


Fig. 4.5 Ejemplo de partición de equivalencia jerárquica

un valor de esa partición. En el caso unidimensional (un dominio y una división), el número mínimo de casos de prueba debería ser tanto como las particiones de equivalencia que hemos identificado. En el caso multidimensional (más de un dominio), el asunto se complica un poco más, ya que el número de casos de prueba dependerá, por ejemplo, de la forma en que tratemos las combinaciones de particiones no válidas, así como de posibles dependencias o restricciones entre valores y particiones de diferentes dominios. La cobertura se mide como el número de particiones de equivalencia probadas utilizando al menos un valor dividido por el número total de particiones de equivalencia definidas y generalmente se expresa como un porcentaje.

Ejemplo Consideremos nuevamente el sistema de verificación PESEL. Supongamos que la partición se parece a la de la Fig. 4.5 y que hemos definido los siguientes casos de prueba:

- PESEL de un adulto nacido en 1898 •
- PESEL de un adulto nacido en 1999 •
- PESEL = "1234" (demasiado corto)

Este conjunto de tres casos de prueba cubre 3 de las 12 particiones de equivalencia identificadas (ver Fig. 4.5), por lo que logra una cobertura de $3/12 = 1/4 = 25\%$.

Cubrir múltiples particiones de equivalencia simultáneamente: enmascaramiento de defectos Se necesita un enfoque ligeramente diferente en una situación en la que nuestras pruebas deben cubrir simultáneamente particiones de equivalencia derivadas de más de un dominio. De tal

En una situación, es una buena práctica no crear casos de prueba que cubran dos o más particiones no válidas. Esto tiene que ver con el llamado enmascaramiento de defectos. La estrategia recomendada es la siguiente:

1. Primero, cree la menor cantidad posible de casos de prueba compuestos únicamente de datos de prueba de particiones válidas, que cubrirán todas las particiones válidas de todos los dominios.
2. Luego, para cada partición no válida descubierta, cree un caso de prueba separado en el que se producirán los datos de esa partición y todos los demás datos provendrán de las particiones válidas.

Considere el siguiente ejemplo para ilustrar este enfoque.

Ejemplo El sistema otorga una calificación a un estudiante basándose en dos datos: el número de puntos de los ejercicios (0–50) y del examen (0–50). El alumno aprueba la asignatura si la puntuación total supera 50. Así, los datos de entrada del caso de prueba constarán de dos partes: puntos por los ejercicios y puntos por el examen. Supongamos que hemos distinguido los siguientes dominios y sus particiones:

Dominio de la variable A = “puntos de ejercicio”:

• (A1) partición no válida: números menores a 0 • (A2) partición válida: números del 0 al 50 • (A3) partición no válida: números mayores a 50

Dominio de la variable B = “puntos del examen”:

• (B1) partición no válida: números menores a 0 • (B2) partición válida: números del 0 al 50 • (B3) partición no válida: números mayores a 50

Queremos cubrir todas las particiones de equivalencia de los dos dominios, A y B. Las particiones válidas son A2 y B2. Todas las demás (A1, A3, B1, B3) son particiones no válidas. Cada caso de prueba contiene como datos de entrada una cierta cantidad de puntos de ejercicio y una cierta cantidad de puntos de examen, por lo que un solo caso de prueba siempre cubre una partición del dominio A y otra del dominio B. Siguiendo la estrategia descrita anteriormente, cubrimos primero las particiones válidas. Como sólo tenemos una partición válida en cada dominio, un caso de prueba es suficiente, por ejemplo:

TC1: A = 25, B = 30 (cubre la partición válida A2 y la partición válida B2).

Quedan dos particiones no válidas para cubrir desde A y dos desde B. Por lo tanto, necesitamos cuatro casos de prueba más en los que estas particiones se probarán individualmente (la otra partición en un caso de prueba determinado debe ser la partición válida):

TC2: A = -8, B = 35 (cubre la partición inválida A1; cubre adicionalmente B2), TC3: A = 48, B = -11 (cubre la partición inválida B1; cubre adicionalmente A2), TC4: A = 64, B = 4 (cubre la partición inválida A3; cubre adicionalmente B2), TC5: A = 12, B = 154 (cubre la partición inválida B3; cubre adicionalmente A2).

Si estuviéramos probando una situación en la que ambos valores provienen de particiones no válidas, podría ocurrir el fenómeno de enmascaramiento de defectos. Supongamos que el sistema verifica que un

El estudiante ha aprobado una materia mediante el siguiente procedimiento (descrito en pseudocódigo):

```

ENTRADA: EjerciciosPuntos, ExamenPuntos
SI (Puntos de ejercicios + Puntos de examen > 50) ENTONCES
    VOLVER "Curso aprobado."
DEMAS
    VOLVER "Curso fallido".
  
```

Ahora considere el siguiente caso de prueba:

TC6: A = -28, B = 105 (cubre las particiones A1 y B3 no válidas).

En esta situación, la puntuación total será $-28 + 105 = 77$ y, por lo tanto, el sistema devolverá un resultado de "Curso aprobado", ¡a pesar de que ambas entradas son incorrectas!

Cobertura de "cada elección"

La cobertura de "cada elección" se aplica al caso multidimensional, es decir, el caso en el que hay más de un dominio y cada caso de prueba cubre una partición de la distribución de cada dominio. Esta cobertura es uno de los tipos de cobertura más simples (y débiles) aplicados al caso multidimensional. Requiere que cada partición de cada dominio se pruebe al menos una vez. En la práctica, al utilizar este método, el evaluador intenta que el siguiente caso de prueba cubra tantos elementos de cobertura previamente descubiertos como sea posible.

Ejemplo Estamos probando un producto COTS para venta general. Esto significa que necesitamos probarlo en diferentes entornos. El programa funciona con diferentes sistemas operativos y diferentes navegadores. Por tanto, es necesario probar el funcionamiento de diferentes navegadores en diferentes sistemas operativos. Supongamos que tenemos los siguientes cuatro navegadores para probar:

- Google Chrome (GC) •
- Firefox (F) •
- Safari (S) •
- Opera (O)

y los siguientes tres sistemas operativos:

- Windows (W) •
- Linux (L) • iOS

Para simplificar, no incluimos versiones específicas de sistemas operativos o navegadores. Cada tipo de navegador forma una partición de equivalencia de un elemento, lo que da como resultado cuatro particiones: {GC}, {F}, {S} y {O}. Cada sistema operativo, a su vez, crea una partición de equivalencia de un elemento, haciendo un total de tres particiones: {W}, {L} y {iOS}.

Según el criterio de cobertura de "cada elección", para cada partición identificada, debe haber un caso de prueba que cubra un valor de esa partición. En nuestro ejemplo, los casos de prueba están representados por un par de datos de prueba (tipo de navegador, sistema operativo).

En nuestro ejemplo, cuatro casos de prueba son suficientes para cumplir el criterio de cobertura, por ejemplo:

TC1: (GC, G)

TC2: (F, L)

TC3: (S, iOS)

TC4: (O, M)

Tipos de problemas detectados La

técnica EP identifica problemas resultantes de un procesamiento de datos defectuoso, es decir, resultantes de errores en el modelo de dominio.

Definir el dominio es clave La técnica

EP siempre se aplica a un dominio específico que modela el problema. A veces, la elección del dominio es obvia, pero otras veces, el asunto puede resultar un poco más complicado. Considere el siguiente ejemplo bastante típico.

Ejemplo. El termostato apaga la calefacción cuando la temperatura (calculada en grados completos) supera los 21 grados y se enciende cuando la temperatura baja de los 18 grados. Diseñe casos de prueba utilizando particiones de equivalencia.

¿Cómo aplicar la técnica EP a este problema? ¿Cuántas particiones de equivalencia habrá que comprobar? Eso depende de qué propiedad específica del sistema queremos verificar. Podemos abordar nuestro problema al menos de tres maneras: Método 1. Analizando solo el dominio, notamos que para los valores 22 y superiores, el sistema apaga la calefacción; para valores 17 y menores, se enciende; y para valores entre 18 y 21, no realiza ninguna acción. Por lo tanto, tenemos tres particiones de equivalencia para el dominio de "temperatura", hasta 17 grados, de 18 a 21 grados y por encima de 21 grados (ver Fig. 4.6a), y para cubrirlas, necesitamos tres casos de prueba, por ejemplo:

- Temperatura = 15 (potencia esperada: calefacción encendida).
- Temperatura = 20 (salida esperada: el estado de calentamiento no cambia en comparación con el estado anterior).
- Temperatura = 22 (salida esperada: calefacción apagada).

Método 2. Tenga en cuenta que en el rango de temperatura 18-21, la calefacción puede estar encendida o apagada. Por tanto, consideramos un dominio compuesto por pares (temperatura, estado de calentamiento). Ante esta situación tenemos cuatro posibilidades:

- Temperatura <18, calefacción encendida
- Temperatura 18–21, calefacción encendida

(continuado)

- Temperatura 18–21, calefacción apagada •

Temperatura >21, calefacción apagada

Así pues, tenemos cuatro situaciones que cubrir (véase la figura 4.6b). La situación ahora es un poco más complicada que antes, porque antes de ejecutar la prueba, debemos forzar la condición de calentamiento adecuada para temperaturas de 18 a 21°C. Por lo tanto, los casos de prueba que cubren las cuatro particiones mencionadas anteriormente podrían verse así:

- TC1: temperatura = 15 (potencia esperada, calefacción encendida). • TC2: temperatura = 18 después de aumentar desde 17 (potencia esperada, la calefacción aún está en funcionamiento). en).
- TC3: temperatura = 21 después de bajar de 22 (potencia esperada, la calefacción es todavía apagado).
- TC4: temperatura = 22 (potencia esperada, calefacción apagada).

Método 3: No podemos considerar pares estáticos (temperatura, estado de calentamiento), sino transiciones entre dichos pares. Entonces nuestro dominio describirá posibles transiciones entre pares (temperatura, estado de calentamiento) y constará de seis elementos posibles (en la siguiente lista de elementos del dominio, los números denotan temperaturas, y APAGADO y ENCENDIDO denotan calentamiento apagado y calentamiento encendido, respectivamente):

- (17, ENCENDIDO) → (18, ENCENDIDO) • (18, ENCENDIDO) → (17, ENCENDIDO) • (18, APAGADO) → (17, ENCENDIDO) • (21, ENCENDIDO) → (22, APAGADO) • (21, APAGADO) → (22, APAGADO) • (22, APAGADO) → (21, APAGADO)

Esta situación se muestra en la figura 4.6c. Por lo tanto, necesitamos seis casos de prueba para simular estas transiciones; por ejemplo, para el primer elemento, la configuración inicial del sistema es de 17 grados y el estado de calentamiento está activado. La prueba consiste en aumentar la temperatura a 18 grados y ver si se apaga la calefacción.

Tenga en cuenta que podríamos agregar cuatro transiciones más a nuestra lista, que implican permanecer en el mismo dominio, si no queremos limitarnos a cambios que involucran transiciones entre diferentes rangos de temperatura que forman particiones de equivalencia del dominio de “temperatura”:

- (17, ENCENDIDO) → (16, ENCENDIDO) • (19, ENCENDIDO) → (20, ENCENDIDO) • (19, APAGADO) → (18, APAGADO) • (22, APAGADO) → (23, APAGADO)

Entonces, como puede ver en el ejemplo anterior, el objetivo de la prueba afecta significativamente la forma del dominio. En nuestro caso, el dominio se modeló en tres

(continuado)


diferentes maneras: como un conjunto de números que representan temperaturas, como un conjunto de pares (temperatura, estado de calentamiento) y como un conjunto de transiciones entre dichos pares. La aplicación de la técnica EP a cada uno de estos resultó en la verificación de un aspecto significativamente diferente del sistema. Por lo tanto, siempre necesitamos saber exactamente cuál es la forma del dominio en el que trabajaremos antes de aplicar esta técnica.

Tenga en cuenta también que, en ocasiones, la necesidad de especificar con precisión el resultado esperado de la prueba nos permite detectar errores o ambigüedades en la especificación. En el método 1 anterior, para los datos de la partición 18 a 21, no está muy claro cuál se supone que es el resultado esperado: si se supone que el termostato debe estar encendido o apagado. Esto puede indicar que el problema que intentamos solucionar está mal planteado.

Al final del análisis de este ejemplo, observemos una cosa más muy importante. Es decir, para este problema particular, la técnica de partición de equivalencia no es una buena opción (los problemas que encontramos al aplicarla lo ilustran muy bien). Esto se debe a que el PE se centra en detectar problemas en la implementación del dominio (estático), y la cuestión fundamental en el problema de prueba que estamos considerando aquí es la verificación del comportamiento del termostato. Parece que una técnica más apropiada sería la prueba de transición de estado (ver Sección [4.2.4](#)).

4.2.2 Análisis de valor límite (BVA)

BVA como extensión del análisis de valores

límite EP es una técnica basada en la  técnica de partición de equivalencia. Por lo que su versatilidad y el tipo de problemas detectados serán similares a este último. La diferencia con la partición de equivalencia es que en BVA seleccionamos elementos muy específicos de las particiones de equivalencia para realizar pruebas, es decir, aquellos que se encuentran en los límites de estas particiones.

Valores límite: ¿a qué dominios se aplica la BVA?

Un valor límite de una partición es el elemento más pequeño o más grande de esa partición.

Hablar de "más pequeño" y "más grande" sólo tiene sentido si definimos una relación de orden entre los elementos del dominio. Por lo tanto, la técnica BVA sólo se puede aplicar a dominios cuyos elementos están ordenados en términos de alguna relación de orden (por ejemplo, a conjuntos de números, con una relación " $<$ "). Ejemplos de tales dominios son conjuntos de números naturales, enteros o reales, valores relacionados con la hora o la fecha, etc.

Los valores límite siempre se definen para una partición de equivalencia específica. Por ejemplo, si el dominio "edad", que contiene los números naturales entre 1 y 120, se divide en dos particiones de equivalencia, $\{1, 2, \dots, 18\}$ (niño) y $\{19, 20, \dots, 120\}$ (adulto), entonces los valores límite de la partición "secundaria" son 1 (la más pequeña) y 18 (la más grande). Los valores límite para la partición "adulto" son 19 y 120.

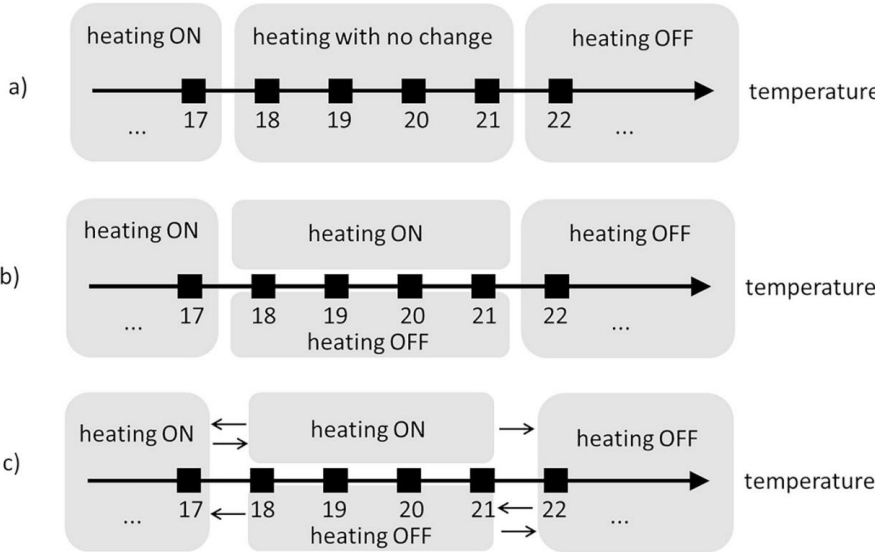


Fig. 4.6 Tres enfoques diferentes para seleccionar un dominio para particiones de equivalencia

Además, debemos suponer una cosa más: las particiones de equivalencia consideradas no pueden tener “huecos”, es decir, formalmente hablando, si dos valores a y b tales que $a < b$ pertenecen a la misma partición de equivalencia, entonces todos los elementos intermedios c , es decir, tal que $a < c < b$, también debe pertenecer a la misma partición. ¿Por qué? Considere el ejemplo mencionado anteriormente pero elimine el elemento 4 de la partición “secundaria”. Esta partición tiene la forma $\{1, 2, 3, 5, 6, \dots, 18\}$. Sus valores extremos siguen siendo, por supuesto, 1 y 18, pero surge la pregunta: ¿no son los valores 3 y 5 también “límites” de esta partición? Después de todo, ¡ambos están adyacentes a un elemento que no pertenece a la partición! En este caso, entonces, tendríamos que dividir nuestra partición en dos, $\{1, 2, 3\}$ y $\{5, 6, \dots, 18\}$, y el valor de 4 sería una partición separada. Entonces, el dominio se dividiría en cuatro particiones: $\{1, 2, 3\}$, $\{4\}$, $\{5, 6, \dots, 18\}$ y $\{19, \dots, 120\}$.

Derivar casos de prueba con BVA

El procedimiento general para derivar casos de prueba utilizando la técnica BVA es el siguiente:

1. Identifique el dominio que desea analizar.
2. Realice la partición de equivalencia de este dominio en particiones de equivalencia.
3. Para cada partición de equivalencia identificada, determine sus valores límite (nota: a veces el análisis puede limitarse solo a ciertas particiones y es posible que no tenga que tener en cuenta todas las particiones de equivalencia determinadas).
4. Para cada valor límite, determine los elementos de cobertura (los elementos que se probarán) para este valor límite.

Los dos primeros pasos simplemente aplican la técnica EP. En el paso 3, identificamos los límites de las particiones. En el paso 4, con base en los valores límite identificados,

determinar los elementos que se utilizarán en los casos de prueba como datos de entrada de prueba. En la técnica BVA, los valores límite (condiciones de prueba) no son necesariamente los mismos que los elementos que se seleccionarán para la prueba (elementos de cobertura). Esto dependerá de qué particiones estemos considerando y de la variante elegida del método BVA. Esto se debe a que existen dos variantes principales del BVA: los llamados BVA de 2 valores y BVA de 3 valores. Discutámoslos en detalle.

BVA de 2 valores

En el BVA de 2 valores [10, 43], para cada valor límite identificado, ese valor y su vecino más cercano que no pertenece a la partición a la que pertenece el valor límite se seleccionan para la prueba. Por ejemplo, si consideramos los límites de la partición $P = \{1, 2, 3, 4, 5, 6\}$ en el dominio de los enteros (es decir, los valores 1 y 6), seleccionamos para probar:

• Valor de 1 (como valor límite de P) • Valor de 0
(como el vecino más cercano de 1 que no pertenece a P) • Valor de 6 (como
valor límite de P) • Valor de 7 (como el vecino
más cercano de 6 no perteneciente a P)

BVA de 3 valores

En el BVA de 3 valores [45, 49] para cada valor de límite identificado, tomamos ese valor y sus dos vecinos para realizar pruebas, independientemente de a qué particiones pertenezcan. En el ejemplo anterior, seleccionaríamos para probar:

• Valor de 1 (como valor límite de P) • Valor de 0
(como vecino izquierdo de 1) • Valor de 2 (como
vecino derecho de 1) • Valor de 6 (como valor
límite de P) • Valor de 5 (como vecino izquierdo
de 6) • Valor de 7 (como vecino derecho de 6)

Comparación de BVA de 2 y 3 valores

Esquemáticamente, el principio de determinación de los valores límite, junto con los elementos de cobertura asociados, se muestra en la Fig. 4.7. Un caso especial puede ser una partición de un solo elemento, pero el principio funciona de manera análoga. El único elemento de esta partición es tanto su elemento más pequeño como su elemento más grande. En el ejemplo de la figura, la partición 3 contiene solo el valor 7. En el caso del BVA de 2 valores, los valores tomados para la prueba serían los siguientes:

• Un valor de 7 como valor de límite mínimo y un valor de 6 como vecino fuera de la partición. • Un
valor de 7 como valor
de límite máximo y un valor de 8 como vecino fuera de la partición.

Entonces tomaríamos el conjunto de valores 6, 7 y 8 para realizar la prueba. De manera similar, para el BVA de 3 valores, los valores para la prueba son 6, 7 y 8 determinados por 7 como valor mínimo y los mismos valores (6, 7, 8) determinados por 7 como valor máximo. Por lo tanto, para la prueba, tomaríamos un conjunto de los mismos valores que en el BVA de 2 valores: 6, 7 y

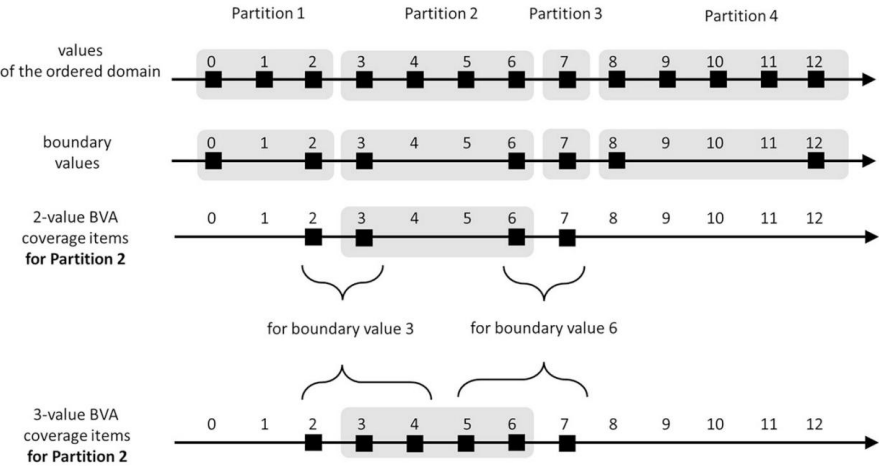


Fig. 4.7 Determinación de límites y elementos de cobertura en las versiones de dos y tres puntos del método AWB

8. Este es el caso sólo para particiones de un solo elemento. Para particiones con al menos dos elementos, el BVA de 3 valores siempre nos dará más elementos para probar que el BVA de 2 valores.

Consideremos ahora algunos ejemplos de la aplicación de la técnica BVA.

Ejemplo El sistema asigna un descuento en el billete a pasajeros menores de 18 años (descuento niños) y mayores de 65 años (descuento senior). Un pasajero entre 18 y 65 años no tiene derecho a descuento. Queremos comprobar la exactitud de la asignación de

el descuento. Llevemos a cabo el procedimiento de cuatro pasos descrito anteriormente para identificar los elementos de cobertura.

Paso 1. Identificar el dominio. La variable a analizar es la edad del pasajero, que es un número entero no negativo. Por tanto, el dominio tiene la forma $\{0, 1, 2, 3, \dots\}$.

Paso 2. Identificar particiones de equivalencia. A partir de la especificación identificamos las siguientes particiones de equivalencia:

- P1: edad elegible para descuento para niños $\{0, 1, 2, \dots, 17\}$
- P2: edad elegible para un boleto regular $\{18, 19, \dots, 64, 65\}$
- P3: edad elegible para adulto mayor descuento $\{66, 67, 68, \dots\}$

Paso 3. Identificar los valores límite:

- Los valores límite de P1 son 0 y 17.
- Los valores límite de P2 son 18 y 65.
- El valor límite de P3 es 66 (en este ejemplo, P3 no tiene el valor límite más grande valor).

Paso 4. Identificar los valores a probar. En caso de que queramos aplicar el BVA de 2 valores para todas las particiones de equivalencia, de hecho, se deben tomar todos los valores límite identificados para realizar la prueba, y eso es suficiente. Esto se debe a que tenemos una especie de simetría entre dos valores de límite adyacentes cualesquiera de dos particiones. Por ejemplo, 65 como valor límite de P2 tiene un vecino 66 desde fuera de la partición, que también es un valor límite de P3, y su vecino de otra partición es 65. Por lo tanto, utilizando el BVA de 2 valores, para realizar pruebas, seleccione todos los valores límite identificados: 0, 17, 18, 65 y 66. Asumimos aquí que el valor -1 no es factible.

En el caso del BVA de 3 valores, elegimos para probar 0, 1, 16, 17, 18, 19, 64, 65, 66 y 67 porque:

- Para el valor límite 0, tomamos este valor y su vecino: 0 y 1 (asumimos que el valor -1 es inviable).
- Para el valor límite 17, tomamos este valor y sus vecinos: 16, 17 y 18.
- Para el valor límite 18, tomamos este valor y sus vecinos: 17, 18 y 19.
- Para el valor límite 65, tomamos este valor y sus vecinos: 64, 65 y 66.
- Para el valor límite 66, tomamos este valor y sus vecinos: 65, 66 y 67.

Por supuesto, algunos valores se repiten, pero tomamos cada valor para probarlo solo una vez. Si solo la partición del medio ("boleto normal") estuviera sujeta a nuestro análisis, nos limitaríamos a identificar los límites de esta partición únicamente, por lo que los valores 18 y 65.

Entonces, en el BVA de 2 valores, elegiríamos los valores 17, 18, 65 y 66 para realizar la prueba, y en el BVA de 3 valores, elegiríamos los valores 17, 18, 19, 64, 65 y 66.

Ejemplo En algún lugar del código, se toma una decisión dependiendo del valor de la variable entera x. La decisión debe ser de la forma:

```
Si x < 16 ENTONCES
    Realizar la ACCIÓN 1
DEMÁS
    Realizar la ACCIÓN 2
```

En este caso existen dos particiones de equivalencia: la primera contiene valores que provocan la ejecución de la "ACCIÓN 1", que son números enteros menores a 16; el segundo es su complemento, es decir, los valores mayores o iguales a 16. Provocan la ejecución de la "ACCIÓN 2". El límite de la primera partición es 15 y el límite de la segunda partición es 16. La variable x no tiene ni el valor más pequeño ni el más grande, por lo que la primera partición no tiene un valor de límite mínimo y la segunda partición no tiene un valor de límite máximo. Así, en el BVA de 2 valores elegiremos 15 y 16 para realizar la prueba, y en el BVA de 3 valores elegiremos 14, 15, 16 y 17.

En la práctica, normalmente existen los valores más grandes y más pequeños a nivel global: estos son, por ejemplo, los rangos aceptados por campos del tipo correspondiente (por ejemplo, un campo acepta un número que consta de hasta cinco dígitos) o los valores de las variables correspondientes. Por ejemplo, una variable de tipo int (entero) en C++ tiene un rango de -231 a 231 - 1, es decir, de -2.147.483.648 a 2.147.483.647. Por lo tanto, un buen evaluador comprobará no sólo los límites de partición establecidos por la especificación sino también los límites establecidos por la arquitectura de la solución bajo prueba, por ejemplo, el rango de valores variables.

Determinación cuidadosa de los valores límite En el caso

del BVA, se debe tener mucho cuidado acerca de cómo se definen los límites de la partición.

Supongamos que estamos trabajando en el dominio de los números naturales. Entonces, por ejemplo, la formulación "la partición contiene elementos no menores a 7" significa que el número más pequeño que pertenece a esta partición es 7. A su vez, la formulación "la partición contiene elementos mayores a 7" significa que el valor más pequeño de esta la partición es 8. De manera similar, "como máximo 65" significa números hasta 65 inclusive, y la condición " $x < 65$ " significa que el valor más grande que satisface esta desigualdad es 64. La frase "la partición contiene los valores que van desde x hasta y" significa que la partición contiene los valores de x y, incluidos x y. Quienes tengan dudas al respecto se animan a plantearse el problema: ¿qué días abre la tienda, en el que cuelga un papel que dice: "la tienda abre de lunes a viernes?". ¿La tienda está abierta 5 días a la semana de lunes a viernes o solo de martes a jueves?

Aplicación El

método BVA es muy simple pero extremadamente efectivo para detectar tipos específicos de defectos. La razón de esto es que los desarrolladores a menudo cometen lo que se llama "errores por uno", implementando incorrectamente los límites de partición de equivalencia. A continuación se muestran dos ejemplos de errores típicos de los desarrolladores que provocan defectos detectables por la BVA:

- El desarrollador debería haber implementado la condición "si $x < 10$ " (desigualdad estricta) pero la implementó erróneamente como "si $x \leq 10$ " (desigualdad débil). • El desarrollador debería inicializar la variable de control del bucle (iterador) como "para $i=0$ a 10", pero asumió erróneamente que los elementos de la matriz están indexados desde 1 y lo escribió como "para $i=1$ a 10".

En el primer caso, las particiones de equivalencia, según la especificación, deberían quedar así: {..., 8, 9}, {10, 11, ...}. Sin embargo, la implementación incorrecta dividió el dominio (debido al flujo de control realizado) de la siguiente manera: {..., 9, 10}, {11, 12,...}. Analizando este ejemplo con el BVA, vemos que los valores límite (válidos, según la especificación) son 9 y 10. Si utilizamos el BVA de 2 valores, tomamos estos mismos valores para probar. En el caso del valor 10, según la especificación, el programa debería tomar el camino correspondiente a la rama "falso" en la decisión "si $x < 10$ ", pero en la implementación incorrecta, la decisión " $x \leq 10$ " es cierto, por lo que el control del programa tomará el camino equivocado. Existe una buena posibilidad de detectar este defecto con un caso de prueba que utiliza 10 como valor de datos de prueba para x.

El BVA de 3 puntos es más fuerte que el BVA de 2 puntos, es decir, hay situaciones en las que el BVA de 2 valores no tiene posibilidades de detectar una falla, mientras que el BVA de 3 valores puede provocar una falla. Considere el siguiente ejemplo.

Ejemplo Un desarrollador implementa un componente para controlar la temperatura en una sala de laboratorio. La temperatura, medida en grados completos, no debe exceder los 10 °C. Cuando se supera este umbral, se activa el mecanismo de enfriamiento. La lógica de negocio es la siguiente:

Si la temperatura no supera los 10°C ENTONCES

Hacer nada

DEMÁS

Encienda el enfriamiento

El desarrollador ha implementado la decisión correcta IF $x \leq 10$ THEN correctamente, ... incorpora- como IF $x == 10$ THEN.... Según la especificación, la partición del dominio debe ser {..., 9, 10}, {11, 12, ...}. Sin embargo, según la implementación incorrecta, la partición es la siguiente, {..., 8, 9}, {10}, {11, 12, ...}, con la partición del medio dando el valor de verdad y las partes izquierda y particiones correctas que dan el valor falso de la decisión en la implementación incorrecta.

Si utilizamos el BVA de 2 valores, identificamos los valores 10 y 11 como valores límite y solo los probamos. Para el valor 10 tanto en la implementación esperada como en la incorrecta, la decisión es verdadera: porque es cierto que $10 \leq 10$ y que $10 == 10$.

En cambio, para el valor 11, ambas decisiones son falsas: pues no es cierto que $11 \leq 10$, y no es cierto que $11 == 10$. Por lo tanto, ninguna de estas dos pruebas podrá detectar el defecto. — el flujo de control del programa (¡que contiene la implementación incorrecta de la decisión!) en ambos casos irá por el camino correcto: ¡el que resulta de la implementación esperada! Esto se debe a que a 10 °C el programa no realizará ninguna acción y a 11 °C encenderá el enfriamiento. Por tanto, el BVA de 2 valores no podrá detectar un defecto en el código.

Sin embargo, si utilizamos el BVA de 3 valores, tendremos que tomar cuatro valores para probar: 9, 10, 11 y 12. En particular, el valor de 9 diferencia las rutas de ejecución según la implementación correcta e incorrecta: en implementación correcta, la decisión $9 \leq 10$ es verdadera y, en caso de implementación incorrecta, la decisión $9 == 10$ es falsa. Por lo tanto, existe la posibilidad de que para $x = 9$, detectemos una operación incorrecta del programa debido a la selección de la ruta de flujo de control incorrecta. Al fin y al cabo, según las especificaciones, a 9 °C el programa no debe realizar ninguna acción, pero activará la refrigeración. Las consideraciones anteriores se resumen en la Tabla 4.3.

Valores fuera del dominio

Finalmente, consideremos un problema más. En el ejemplo con la asignación de un descuento en el boleto, indicamos que para un valor límite de 0, no tomamos su vecino -1 para probar, porque asumimos que es inviable, es decir, el usuario no puede ingresar información incorrecta, valores negativos. Sin embargo, en la práctica esto no tiene por qué ser siempre así. Si, por ejemplo, el usuario ingresa un valor de edad en un campo de formulario desde el teclado, puede, por supuesto, ingresar un valor negativo. La prueba o no de valores límite extremos fuera del dominio depende de si la interfaz lo permite. Si es así, el evaluador, por supuesto, debería realizar la prueba.

Cobertura

Los elementos de cobertura en el BVA son los valores límite de las particiones de equivalencia (en el BVA de 2 valores) o los valores límite junto con todos sus valores vecinos (en el BVA de 3 valores). Por lo tanto, para BVA de 2 valores, la cobertura se define como el cociente del número de valores límite probados y el número total de valores límite identificados. Para el BVA de 3 valores, es el cociente del número de pruebas

Tabla 4.3 Diferencias entre BVA de 2 y 3 valores

	BVA de 2 valores		BVA de 3 valores			
	Valores límite identificados (y valores para prueba): x = 10 x = 11					
			x = 9	x = 10	x = 11	x = 12
	Resultados de la prueba					
Especificación: x ≤ 10	(10 ≤ 10) VERDADERO	(11 ≤ 10) FALSO	(9 ≤ 10) VERDADERO	(10 ≤ 10) VERDADERO	(11 ≤ 10) FALSO	(12 ≤ 10) FALSO
Implementación incorrecta: x = 10	(10 = 10) VERDADERO	(11 = 10) FALSO	(9 = 10) FALSO	(10 = 10) VERDADERO	(11 = 10) FALSO	(12 = 10) FALSO
Resultado:	Prueba superada superada	prueba	Prueba fallida	Prueba aprobada	Prueba aprobada	Prueba aprobada
Interpretación:	Ambas pruebas pasaron, defecto no detectado		Hay una prueba que detectó el defecto.			

valores de límite con sus vecinos y el número total de valores de límite identificados y sus vecinos. Esto se debe a que en el BVA de 3 valores, no solo tomamos valores límite para realizar pruebas, sino también valores del interior de las particiones. Por ejemplo, para la partición {2, 3, 4, 5, 6}, los valores tomados para la prueba en el BVA de 3 puntos serán, en particular, los números 3 y 5, que no son los valores límite para esta partición. .

4.2.3 Prueba de la tabla de decisiones

La prueba de la tabla de decisiones de aplicaciones es una técnica que se utiliza para verificar la exactitud de las implementaciones de reglas comerciales. Una regla de negocio suele adoptar la forma de una implicación lógica:

SI (condición) ENTONCES (acción),

donde tanto la condición como la acción pueden estar compuestas por más de un factor. Entonces estamos ante una combinación de condiciones o acciones. A continuación se muestran algunos ejemplos de reglas comerciales:

- SI (edad del cliente < 18) ENTONCES (asignar un descuento en el billete);
- SI ($a+b>c>0$ AND $a+c>b>0$ AND $b+c>a>0$) ENTONCES (puedes construir un triángulo con lados de longitud a, b, c);
- SI (Salario mensual > 10000) ENTONCES (otorgar préstamo bancario Y ofrecer un tarjeta dorada).

Las tablas de decisión nos permiten probar sistemáticamente la corrección de la implementación de combinaciones de condiciones. Este es uno de los llamados combinatorios.

Tabla 4.4 Ejemplo de tabla de decisiones

	Reglas comerciales 1 a 8							
	1	2	3	4	5	6	7	8
Condiciones								
¿Tiene una tarjeta de fidelización?	SI	SI	SI	SI	NO	NO	NO	NO
¿Cantidad total > \$1000?	SÍ	SÍ	NO	NO	SÍ	SÍ	NO	NO
¿Compras en los últimos 30 días? SÍ NO	SI	NO	SI	NO	SI	NO	Acciones	
Descuento concedido	10%	5%	5%	0%	0%	0%	0%	0%

técnicas. Para obtener más información sobre estas técnicas, consulte el programa de estudios Nivel avanzado: Analista de pruebas [28].

Construcción de la tabla de decisión

Describiremos la construcción de la tabla de decisión usando un ejemplo (ver Tabla 4.4).

La tabla de decisiones consta de dos partes, que describen las condiciones (parte superior) y las acciones (parte inferior), respectivamente. Las columnas individuales describen reglas comerciales. La Tabla 4.4 describe las reglas para asignar un descuento en compras en función de tres factores que describen al cliente en cuestión:

- ¿Tiene el cliente una tarjeta de fidelización? (SÍ o NO) • ¿El monto total de las compras excede los \$1000? (SI o NO) • ¿El cliente ha realizado compras en los últimos 30 días? (Sí o no)

Según las respuestas a estas preguntas, se asigna un descuento: 0%, 5% o 10%.

Ejemplo Un cliente tiene una tarjeta de fidelización y hasta el momento ha realizado compras por valor de 1250 \$, y las últimas compras se realizaron hace 5 días. Esta situación corresponde a la regla 1 (tiene tarjeta de fidelización, monto total >\$1000, compras en los últimos 30 días). Entonces, el sistema debería asignar un descuento del 10% a este cliente.

Derivar casos de prueba a partir de la tabla de decisiones

El proceso de creación de casos de prueba específicos utilizando tablas de decisión se puede presentar en los siguientes cinco pasos.

Paso 1. Identifique todas las condiciones individuales posibles (a partir de la base de prueba, por ejemplo, basadas en especificaciones, conversaciones con clientes, el llamado sentido común) y enumérelas en las filas consecutivas en la parte superior de la tabla. Si es necesario, divida las condiciones compuestas en condiciones individuales. Las condiciones suelen aparecer en las especificaciones como fragmentos de oraciones precedidos por palabras como "si", "en el caso de que", etc.

Paso 2. Identifique todas las acciones correspondientes que pueden ocurrir en el sistema y que dependen de estas condiciones (también derivadas de la base de prueba), y enumérelas en las líneas consecutivas en la parte inferior de la tabla. Las acciones suelen aparecer en las especificaciones como fragmentos de oraciones precedidos por palabras como "entonces", "en este caso", "el sistema debería", etc.