Engineer

Defecto

deficiencia que impide al producto cumplir sus requisitos

Qué es

Un defecto es algo incorrecto dentro de un producto de trabajo que hace que se comporte de forma distinta a lo previsto o que omita parte de lo que debía hacer. La mayoría de los programadores lo llaman bug, y quienes prueban software y redactan estándares también utilizan fault o flaw.

La palabra importante es *trabajo*. Un defecto puede estar en el código fuente, pero también en un requisito que contradice a otro, un diseño que no soporta la carga para la que se concibió, un archivo de configuración, una prueba que comprueba un valor equivocado o un manual que describe un botón que el producto eliminó hace dos versiones. Dondequiera que esté, el defecto pertenece al propio producto de trabajo y está presente aunque nadie lo haya advertido.

El vocabulario de las pruebas distingue tres palabras cercanas, y la separación resulta útil. Una persona comete un error: interpreta mal una especificación, hace una suposición incorrecta o pulsa una tecla equivocada. El error deja un defecto en lo que esa persona produjo. Cuando la parte defectuosa se ejecuta en condiciones que lo ponen de manifiesto, el producto se comporta incorrectamente y ese comportamiento visible es un fallo. La cadena avanza en una dirección y puede detenerse en cualquier eslabón. Un defecto en una rama que nunca se ejecuta, o que necesita una entrada que nadie envía, puede permanecer años en una base de código sin causar un solo fallo.

Qué se considera incorrecto depende de lo que se esperaba, y las expectativas van más allá de la especificación escrita. El modelo de calidad de producto de ISO/IEC 25010 describe la adecuación funcional como el grado en que un producto «proporciona funciones que satisfacen necesidades expresas e implícitas cuando se utiliza en condiciones especificadas». La divide en

completitud —si las funciones cubren las tareas especificadas y los objetivos de los usuarios previstos—, corrección —si los resultados son exactos— y pertinencia —si las funciones ayudan a completar la tarea—. Un total incorrecto es un problema de corrección; un informe que nadie puede exportar es un problema de completitud. El mismo modelo trata la fiabilidad como una característica distinta, con ausencia de fallos, disponibilidad, tolerancia a fallos y capacidad de recuperación entre sus partes, de modo que un producto que calcula la respuesta correcta y después se cae dos veces al día sigue teniendo defectos.

Una regresión es un defecto con historia: algo que funcionaba y que un cambio posterior rompió. Tiene un nombre propio porque aporta una pista que no tiene un defecto original: la última versión de la que se sabe que funcionaba.

Utilice el concepto en una decisión

Aplique «¿Qué es un defecto?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Un equipo que lleva una idea del boceto a un producto al que sus clientes pueden acceder, sin pasarla entre tres proveedores.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Ingresos multiplicados por 10 con mapas GIS — 10x crecimiento de ingresos impulsado desde el producto cartográfico

Ingresos multiplicados por 10 con mapas GIS Póngase en contacto →

Por qué importa

El coste de un defecto depende sobre todo de dónde se descubre. Si lo detecta quien escribe el código, cuesta un minuto y no deja rastro. Si se detecta en una revisión, cuesta una conversación. Si lo descubre un cliente, cuesta el propio fallo, la incidencia de soporte, la investigación, la corrección y algo de confianza; y si causó un daño real, pasa a formar parte de un incidente. Los sistemas de tipos, analizadores estáticos, pruebas y revisiones de código persiguen el mismo objetivo, intentando acercar el momento del descubrimiento al momento en que se escribió el defecto.

Los defectos también aparecen en familias. El error que hay detrás de uno suele tener parientes: la misma interfaz mal interpretada llamada igual en seis lugares, la misma confusión de unidades copiada en cada módulo nuevo, la misma suposición sin comprobar en cada manejador de peticiones. Después de un defecto, lo más productivo no es tanto preguntar dónde está la línea defectuosa como qué hizo fácil escribirla y en qué otros lugares ocurrió lo mismo.

Además, los defectos vuelven. El código cambia constantemente y un defecto corregido puede regresar con una refactorización, una fusión o una actualización de dependencias. Una prueba escrita para fallar con el defecto y pasar con la corrección convierte esa corrección en una propiedad duradera de la compilación. Si aun así aparece una regresión, el historial de versiones se convierte en un espacio de búsqueda: `git bisect` realiza una búsqueda binaria entre los commits de una versión de la que se sabe que funcionaba y otra de la que se sabe que fallaba, de modo que un intervalo de mil commits se reduce al que introdujo el bug en unas diez ejecuciones de prueba.

Algunos defectos también son problemas de seguridad. Common Weakness Enumeration de MITRE cataloga debilidades, cada una de las cuales describe como «una condición en un componente de software, firmware, hardware o servicio que, en determinadas circunstancias, podría contribuir a la introducción de vulnerabilidades». Muchas entradas de esa lista son defectos corrientes que un atacante puede aprovechar: una comprobación de límites ausente, una entrada del usuario enviada sin escapar a una consulta o un paso de autorización omitido en una ruta.

Cómo se aplica en la práctica

Los defectos más costosos de este sitio fueron aquellos que todas las señales automáticas daban por buenos. Una auditoría descubrió once defectos de accesibilidad que las reglas automatizadas habían dejado pasar, porque cada uno dependía de cómo se representaba una página, en lugar de lo que decía su marcado. Dos estaban en las propias comprobaciones. Todas las ventanas que abría la comprobación de disposición tenían 900 píxeles de alto, así que nunca se había observado un teléfono en horizontal de 852 por 393, con siete píxeles entre dos elementos; y el motor de reglas había estado ejecutando 70 de sus 105 reglas. Cada defecto se incorporó a un registro junto con la regla que originó, de modo que el trabajo cierra una clase de error y no solo un caso. Una vez corregida la propia comprobación de disposición, falló 128 veces en tres idiomas.

Un segundo defecto no produjo síntoma alguno. El sitio tiene un tema oscuro predeterminado y otro claro para los lectores cuyo sistema lo solicita, y su fondo a página completa se construye con tres capas en el tema oscuro. Una pequeña comprobación estructural que cuenta las capas que pinta cada tema descubrió que el tema claro solo dibujaba dos en su primera ejecución. La superposición había estado ausente desde que se escribió el tema claro. Nada emitía un error y

todas las páginas se representaban, y un fondo con una capa menos parece una decisión deliberada, así que durante meses no hubo motivo para mirar. La corrección fue una regla; la comprobación es lo que mantiene la corrección.

Mantener las correcciones es tarea del motor de controles del repositorio, donde una regla vincula cada corrección de bug con pruebas que la respaldan: un commit marcado como corrección tiene que aportar una prueba que falle sin él. Los controles también tienen 146 pruebas propias, porque un control defectuoso puede dejar pasar los cambios que debía rechazar y aparentar que funciona.

Dónde surgen los problemas

Se corrige el caso y la clase sigue abierta. Se corrige la línea señalada, se cierra la incidencia y tres errores de la misma familia siguen esperando en otras partes del código. Después de cada corrección, busque el patrón que produjo el defecto y, donde una máquina pueda reconocerlo, convierta la búsqueda en una comprobación.

Vuelve un bug corregido. Una refactorización medio año después restaura el comportamiento anterior porque nada registró qué protegía la corrección. Escriba la prueba antes de corregir, observe cómo falla con el código antiguo y consérvela para siempre en el conjunto de pruebas.

El comprobador contiene el defecto. Un conjunto de reglas que carga silenciosamente solo una parte de ellas, una lista de ventanas de prueba todas con la misma altura, una aserción que comprueba lo que hace el código en lugar de lo que debería hacer: cada uno informa de éxito mientras el defecto está a la vista. Dé a las comprobaciones sus propias pruebas y lea en la salida de la herramienta qué reglas se ejecutaron realmente.

La cobertura sustituye a la corrección. Un conjunto de pruebas puede ejecutar todas las líneas y aun así aceptar un resultado incorrecto si ninguna aserción examina el resultado, y un código que cumple su especificación escrita puede seguir sin satisfacer una necesidad implícita. Utilice la cobertura para encontrar código que ninguna prueba comprueba y evalúe una prueba por los defectos que detectaría.

«Funciona según el diseño» cierra un informe válido. El comportamiento coincide con la especificación, se rechaza el informe y el producto de trabajo defectuoso había sido la especificación desde el principio. Cuando el problema está en el requisito o en el diseño, registre el defecto en ese requisito o diseño para que la corrección se haga en el origen.

Lecturas adicionales

- Defecto — la definición del ISTQB utilizada en las pruebas de software, con sus sinónimos y los términos relacionados error y fallo.
- ISO/IEC 25010 — una descripción general del modelo de calidad de producto y de las subcaracterísticas de cada característica, entre ellas la adecuación funcional y la fiabilidad.
- git-bisect — el manual de Git para la búsqueda binaria que encuentra el commit que introdujo un bug, incluidas ejecuciones automatizadas con un script de prueba.
- Acerca de CWE — qué es Common Weakness Enumeration, cómo se organiza su lista de debilidades de software y hardware y quién la mantiene.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es un defecto?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/defect/