Materia: Computación Tolerante a Fallas.

NRC: 179961

Maestro: Lopez Franco, Michael Emanuel

Aula: X-02

Sección: D06

Alumno: Zashuvath López Moreno, Ethan Israel

Código: 216493953

Índice

Introducción

Leeremos e investigaremos sobre los diferentes métodos para la prevención de defectos.

Objetivo:

Conocer y entender los métodos de prevención de defectos.

Contenido

Algunos de los métodos mas comunes para la prevención de defectos son muy sencillo e incluso obvios si lo pensamos de forma detenida, a continuación presentare algunos tipos o consejos para llevar a cabo esta prevención de forma óptima.

Revisión de Código: El revisar el código es algo mas que solo verlo, es el acudir con personas que tienen conocimientos sobre estos temas, los mismos buscan errores y posibles problemas, además de asesorar al creador del código para simplificar o mejorar el mismo sin comprometer los resultados.

Pruebas de Software: El poner a prueba el software es simple y sencillamente el someter a diferentes condiciones el software para con esto lograr identificar errores y defectos, además estas pruebas pueden incluir las pruebas de unidad, integración, sistema y aceptación.

Desarrollo Guiado por Pruebas (TDD): En el TDD, se escriben una serie de pruebas automatizadas previas a la escritura del código real, gracias a esto se nos es más facil garantizar el correcto funcionamiento y el cumplimiento d ellos requisitos disminuyendo o eliminando los defectos desde el principio.

Control de Versiones: El usar diferentes sistemas de control de versiones como puede ser Git nos permite realizar un seguimiento a los cambios que sufre el código, esto a su vez nos facilita la detección y corrección de defectos.

Diseño Sólido: Un diseño solido y una arquitectura que este bien planificada puede prevenir muchos problemas antes de que siquiera aparezcan, además utilizar patrones de diseño y una buena practica de arquitectura puede ayudar en ese sentido.

Pruebas de Seguridad: Junto a las pruebas funcionales, las pruebas de seguridad son algo esencial para poder prevenir vulnerabilidades y defectos relacionados con la seguridad en el software.

Revisión de Diseño y Requisitos: Algo que es muy común de pasar por alto es que mucho antes de comenzar a codificar, es indispensable el revisar y validar los diseños y requisitos del programa para con esto asegurarnos de comprender y tener total conocimiento de la idea o problema a solucionar.

Automatización: El proceso de automatización para las tareas repetitivas como pruebas de regresión nos pueden ser muy útiles para identificar de forma rápida defectos que son introducidos durante el desarrollo.

Estándares de Codificación: El estandarizar la codificación y seguir practicas correctas de programación nos evitan muchos errores y defectos que son comunes en el día a día.

Formación y Educación: La formación continua y la educación en desarrollo de software son importantes para que los programadores adquieran habilidades y conocimientos que les ayuden a evitar errores en su trabajo.

Gestión de Proyectos Ágil: Los enfoques ágiles de gestión de proyectos, como Scrum o Kanban, fomentan la colaboración constante, la retroalimentación y la adaptación, lo que facilita la detección y corrección de errores de manera más efectiva.

Herramientas de Análisis Estático: El uso de herramientas de análisis estático de código puede ayudar a identificar posibles errores y problemas de estilo en el código antes de que se compile.

Mentoría y Revisión por Pares: La mentoría y la revisión por pares son prácticas que pueden mejorar la calidad del código, ya que permiten que desarrolladores más experimentados compartan conocimientos y brinden retroalimentación a otros.

Ejemplos

Revisión de Código: Después de escribir una función para calcular el promedio de una lista de números, otro desarrollador revisa el código y señala que falta manejar el caso de una lista vacía, evitando así un posible error en tiempo de ejecución.

Pruebas de Software: Antes de lanzar una nueva versión de una aplicación móvil, un equipo de pruebas ejecuta la aplicación en varios dispositivos y encuentra un error en el botón de inicio que no funciona correctamente.

Desarrollo Guiado por Pruebas (TDD): Un programador utiliza TDD para crear una función de suma en Python. Primero, escribe una prueba que falla (porque la función aún no existe), luego implementa la función de suma para que la prueba pase correctamente.

Control de Versiones: Después de una serie de cambios en el código de un proyecto, un desarrollador introduce un error en una función. Utilizando Git, puede retroceder al estado anterior del código antes de que se introdujera el error.

Automatización: Para asegurarse de que no se introduzcan nuevos errores al agregar código, un equipo configura una suite de pruebas automatizadas que se ejecutan automáticamente cada vez que se realiza una confirmación en el repositorio de código.

Estándares de Codificación: Un equipo de desarrollo acuerda seguir un estándar de codificación que prohíbe el uso de nombres de variables ambiguos. Esto evita problemas futuros relacionados con la legibilidad del código.

Herramientas de Análisis Estático: Antes de enviar el código a revisión, un desarrollador utiliza una herramienta de análisis estático que detecta un posible problema de fuga de memoria en una función y lo corrige.

Mentoría y Revisión por Pares: Un desarrollador más experimentado revisa el código de un colega junior y sugiere una optimización en un bucle que mejora el rendimiento del programa.

Gestión de Proyectos Ágil: Durante una reunión de revisión en un equipo Scrum, el equipo detecta un error de diseño en una característica y lo agrega al backlog para su corrección en el próximo sprint.

Pruebas de Seguridad: Un equipo de seguridad de la información realiza una prueba de penetración en una aplicación web y descubre una vulnerabilidad de inyección SQL, que luego se corrige antes de que pueda ser explotada por un atacante.

Conclusión

El correcto análisis y la elaboración de modelos y estándares a seguir dentro de al ámbito de la programación y codificación son dos puntos primordiales para evitar muchos errores simples que a largo plazo nos complican el cumplimiento de la idea o provocan frustración en los trabajadores, debido a esto es también necesario el tener una serie de parámetros, ideas y herramientas que nos faciliten el análisis y desarrollo de nuestras ideas, para con ello lograr una menor taza de errores o fallos humanos.

Referencias

Atlassian. (s. f.). *Los distintos tipos de pruebas en software | Atlassian*.

https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing

Lee, G. (2023). Tipos de pruebas de software: diferencias y ejemplos. *LoadView*. https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/

Mtp. (2023). Prevención de defectos en el aseguramiento de la calidad software. *MTP*. https://www.mtp.es/blog/testing-software/prevencion-de-defectos-en-el-aseguramiento-de-la-calidad-del-software/

Prevención de defectos. (s. f.). https://es.slideshare.net/Hernan.Ordonez/prevencin-de-defectos

¿Qué es la prueba de software y cómo funciona? | *IBM*. (s. f.). https://www.ibm.com/mx-es/topics/software-testing

Unir, V. (2022, 5 septiembre). La importancia de las pruebas de software. *UNIR*. https://www.unir.net/ingenieria/revista/pruebas-software/