Centro Universitario de Ciencias Exactas e Ingenierías

Asignatura: **Computación tolerante a fallas**

Sección: D06

Ejercicio 4: Principios y prevención de defectos (2)

Alumno: Luis Jaime Portillo Correa

Código: 217546155

Profesor: **Michel Emanuel López Franco**

Fecha: **30/08/2023**

Objetivo

Investiga sobre diferentes métodos para la prevención de defectos.

Desarrollo

Prevención de defectos

En el área del software, la prevención de defectos es el conjunto de prácticas o técnicas que son destinadas a evitar que se introduzcan errores o defectos en el software durante su desarrollo. El objetivo principal de la prevención de defectos es mejorar la calidad del software y reducir los costos asociados con la detección y corrección de errores en etapas posteriores del ciclo de desarrollo.

La prevención de defectos en el software es una parte crucial de la gestión de la calidad del software. Al implementar estas prácticas y técnicas, las organizaciones pueden reducir la cantidad de errores en sus productos de software y mejorar la satisfacción del cliente, la eficiencia del desarrollo y la confiabilidad del software resultante.

Entre algunas de las técnicas de prevención de defectos más comunes, se encuentran:

Revisión de Código: Las revisiones de código son un proceso en el que otros miembros del equipo revisan el código escrito por un desarrollador para identificar defectos y problemas. Esto puede hacerse de manera formal o informal y es efectivo para identificar errores antes de que se conviertan en problemas más grandes.

Pruebas Unitarias: Las pruebas unitarias son pequeñas pruebas automatizadas que verifican el funcionamiento de unidades individuales de código. Estas pruebas se ejecutan de manera continua durante el desarrollo y ayudan a identificar problemas tempranos.

Pruebas de Integración: Estas pruebas se centran en verificar que las diferentes partes del software funcionen bien juntas. Ayudan a detectar problemas que pueden surgir cuando los componentes se integran.

Pruebas de Aceptación del Usuario (UAT): Estas pruebas se realizan con usuarios finales o sus representantes para garantizar que el software cumple con los requisitos y expectativas del usuario.

Análisis Estático de Código: Herramientas de análisis estático de código escanean el código fuente en busca de posibles problemas, como código no utilizado, variables no inicializadas y otras cuestiones de estilo y calidad.

Modelado y Diseño Eficaz: Un diseño bien pensado y modelado adecuadamente pueden prevenir muchos problemas antes de que ocurran. Utilizar técnicas de diseño sólidas y modelado de datos puede ayudar a prevenir defectos.

Desarrollo Guiado por Pruebas (TDD): En TDD, se escriben pruebas antes de escribir el código real. Esto ayuda a los desarrolladores a pensar en los requisitos y la funcionalidad antes de comenzar a codificar y, por lo tanto, reduce la posibilidad de errores.

Gestión de Requisitos Eficiente: Asegurarse de que los requisitos estén bien definidos, documentados y comprensibles puede evitar problemas causados por malentendidos y cambios frecuentes en los requisitos.

Control de Versiones y Gestión de Configuración: Utilizar herramientas de control de versiones y gestionar cuidadosamente la configuración del software puede evitar problemas relacionados con la inconsistencia de versiones y la falta de seguimiento de cambios.

Entrenamiento y Formación: Proporcionar capacitación y formación a los miembros del equipo de desarrollo en las mejores prácticas de programación y en las tecnologías utilizadas puede ayudar a prevenir errores comunes.

Revisiones de Diseño: Antes de comenzar a escribir código, se pueden llevar a cabo revisiones de diseño para asegurarse de que la arquitectura y el diseño del software sean sólidos y cumplan con los requisitos.

Métricas de Calidad de Software: Utilizar métricas de calidad de software puede ayudar a identificar áreas problemáticas y rastrear la mejora continua.

¿Qué es Orthogonal Defect Classification (ODC)

es un marco sistemático para la clasificación de defectos de software desarrollado por IBM a principios de la década de 1990.También es un concepto que permite la retroalimentación en proceso a los desarrolladores mediante la extracción de firmas en el proceso de desarrollo de defectos.

Utiliza información semántica de defectos para extraer relaciones causa-efecto en el proceso de desarrollo, tiene mecanismos incorporados para el escape de fase y el análisis de causa raíz y se conoce como una "MRI" en un defecto de software.

Valores de ODC

Proporcione comentarios rápidos y efectivos a los desarrolladores.

Captura información de defectos que ocurrieron a través de las fases de desarrollo y el uso del campo.

Permite comprender las tendencias de defectos a lo largo de las fases del ciclo de vida debido a la consistencia de los tipos de defectos.

A través de la medición y el análisis multidimensionales, ODC ayuda a los desarrolladores a gestionar adecuadamente sus procesos de desarrollo y la calidad del producto.

Secciones ODC

Sección de apertura: Cuando encuentre un defecto, se pueden clasificar los siguientes atributos:

Actividad: Esta es la actividad real realizada en el momento del descubrimiento de defectos.

Desencadenante: El entorno o condición que tenía que existir para que el defecto apareciera.

Impacto: Para defectos en proceso, seleccione el impacto que juzga que el defecto habría tenido en el cliente si se hubiera escapado al campo.

Sección más cercana: cuando se sabe cómo se solucionó el defecto, se pueden clasificar los siguientes atributos:

Destino: representa la identidad de alto nivel de la entidad que se fijó.

Tipo de defecto: Representa la naturaleza de la corrección real que se realizó.

Calificador (se aplica al tipo de defecto): captura el elemento de una implementación inexistente, incorrecta o irrelevante.

Fuente: Identifica el origen del objetivo (es decir, diseño / código, identificación, etc.) que tenía el defecto.

Edad: Identifica el historial del objetivo (es decir, diseño / código, identificación, etc.) que tenía el defecto.

Conclusión

Es importante conocer y saber para qué sirve la prevención de defectos en el desarrollo de software, ya que con sus métodos podemos garantizar la calidad y eficiencia de los productos y sistemas que desarrollamos.

Una correcta inversión en técnicas de prevención de defectos, tanto monetaria como de tiempo, nos puede generar varias ventajas como ahorro de costos, mejorar la reputación de nuestro equipo de desarrollo, aumentar la productividad, reducir costos relacionados con la seguridad de los datos, etc.

Bibliografía

- Métodos y técnicas de prevención de defectos. (s. f.). Seguimiento De Defectos De Errores. https://spa.myservername.com/defect-prevention-methods#Defect_Prevention_Methods_and_Techniques
- Prevención de defectos. (s. f.). https://es.slideshare.net/Hernan.Ordonez/prevencin-de-defectos
- Software Quality Exp. (2018, 21 marzo). What is Orthogonal Defect Classification (ODC)? By Vivek Vasudeva. Medium. https://medium.com/@SWQuality3/what-is-orthogonal-defect-classification-odc-by-vivek-vasudeva-f2e49917f478