Computación Tolerante a Fallas

Sección D06

Principios de prevención de defectos

Fernández Venegas David Guadalupe

Código: 216437352

Profesor: López Franco Michel Emanuel

Introducción:

En cualquier ámbito, cualquier desarrollo de un proyecto está propenso a toparse con errores, ya sea desde el planteamiento del proyecto, el inicio o el desarrollo de este. Esto debido a que no hay humano exento de cometer errores. Esto provoca un si fin de defectos en cualquier desarrollo llevado a cabo, por lo que es importante ser conscientes de ello para poder abordar dichos problemas, sobre todo antes de que sucedan. Durante este trabajo estaremos explorando algunos de los principios establecidos para la prevención de defectos sobre todo enfocado en el ámbito de la computación y sobre todo en el desarrollo de software.

Desarrollo:

La prevención de defectos es fundamental para garantizar la calidad y fiabilidad del software que se desarrolla. Los defectos, también conocidos como errores o "bugs" pueden tener un impacto significativo en la funcionalidad y seguridad de una aplicación. Para abordar este desafío, se han desarrollado una serie de métodos y enfoques que ayudan a prevenir la aparición de defectos desde las etapas iniciales del ciclo de desarrollo. Estos métodos se centran en la identificación temprana, la corrección oportuna y la mitigación de riesgos asociados con la programación.

Prácticas aplicadas para la prevención de defectos:

- Revisión de código: Los desarrolladores revisan el código de sus colegas en busca de errores, malas prácticas y oportunidades de mejora. Esto se puede hacer de manera formal a través de revisiones planificadas o de manera informal mediante pares de programadores trabajando juntos. Muchos defectos, como pérdidas de memoria, paso incorrecto de argumentos, código inalcanzable, falta de legibilidad, alta complejidad y problemas de mantenimiento, se pueden identificar mediante la revisión del código. Encontrar defectos en la etapa de codificación y solucionarlos allí mismo resultaría menos costoso que encontrarlos en la etapa de prueba.

- Pruebas unitarias: Las pruebas unitarias implican escribir pruebas automatizadas para cada unidad de código (generalmente una función o método) para asegurarse de que funcionen correctamente. La implementación de pruebas unitarias puede ayudar a detectar defectos temprano y garantizar que las modificaciones futuras no rompan el código existente.

- Análisis estático de código: Herramientas de análisis estático, como linters y analizadores de código estático, escanean el código en busca de posibles problemas,

como variables no utilizadas, declaraciones redundantes o prácticas de codificación inconsistentes.

- Buenas prácticas de codificación: Adherirse a buenas prácticas de codificación, como la nomenclatura de variables coherente, la estructuración del código legible y el uso de comentarios descriptivos puede ayudar a prevenir defectos relacionados con la calidad del código.

- Pruebas de integración: Las pruebas de integración verifican la interacción entre diferentes componentes del software para detectar problemas de comunicación o de funcionamiento en conjunto.

- Control de versiones y gestión de cambios: Utilizar sistemas de control de versiones como Git y llevar un registro de los cambios realizados en el código puede ayudar a identificar y solucionar problemas causados por cambios no deseados. Todos los miembros deben revisar los cambios de código con respecto a otros módulos y dar su opinión en caso de que se sospeche algún efecto secundario.

- Revisión de requisitos: Es importante asegurarse de que los requisitos del proyecto estén bien definidos y comprensibles para evitar malentendidos y cambios de último minuto que puedan introducir defectos. Por lo que el primer nivel de revisión debe estar dentro del equipo, seguido de otro nivel de revisión externa para asegurarse de que todas las perspectivas estén sincronizadas.

- Capacitación y formación: Mantener a los desarrolladores actualizados con respecto a las mejores prácticas de programación y las nuevas tecnologías puede mejorar la calidad del código producido.

- Pruebas de regresión: Realizar pruebas de regresión después de realizar cambios en el código para asegurarse de que las nuevas modificaciones no hayan introducido defectos en áreas previamente funcionales.

- Automatización de pruebas: Automatizar las pruebas funcionales y de rendimiento puede ayudar a identificar rápidamente defectos a medida que se desarrolla el software.
- Uso de estándares y directrices: Seguir estándares de codificación y directrices específicas para el lenguaje o el marco que estás utilizando puede ayudar a mantener la consistencia y prevenir defectos.

Conclusión:

La prevención de defectos en el desarrollo de software es un proceso extenso que abarca varios métodos y prácticas. Desde la revisión de código hasta las pruebas unitarias, el análisis estático de código y el control de versiones, cada técnica se encarga de un papel importante para la identificación temprana y la corrección de problemas. Además, la capacitación continua y el seguimiento de estándares de codificación son fundamentales para mantener la calidad del código. Al implementar estas estrategias de manera efectiva, se puede mejorar significativamente la calidad del software y reducir la presencia de defectos, lo que nos lleva a lograr una mejor experiencia para el usuario y a la satisfacción del cliente.

Bibliografía:

- W&B Asset Studio. (2023, 16 noviembre). Aseguramiento de la calidad en el software: ¿qué es y qué hace? | W&B Asset Studio. Exquisite Goods. https://www.wbassetstudio.com/blog/aseguramiento-de-la-calidad-en-el-software-que-es-y-que-hace/#:~:text=clave%20de%20SQA.-,Prevenci%C3%B3n%20de%20defectos%3A%20el%20objetivo%20principal%20del%20SQA%20es%20prevenir,minimizar%20la%20probabilidad%20de%20errores.

- Follow, M. (2020, septiembre 5). Defect prevention methods and techniques. GeeksforGeeks. https://www.geeksforgeeks.org/defect-prevention-methods-and-techniques/

- de la Plaza, J., & Martín, M. (2018, abril). *Prevención de Defectos en el aseguramiento de la calidad del software, más vale prevenir que curar*. MTP, Digital business assurance. https://www.mtp.es/blog/testing-software/prevencion-de-defectos-en-el-aseguramiento-de-la-calidad-del-software/#Ejemplos_reales_de_defectos_analizados_en_el_sector_Telco

- Khan, A. K. (2013). Software Excellence Augmentation through Defect Analysis and Avoidance. Science, Technology and Arts Research Journal, 2(1), 64-68.