CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENERIAS

Aldo Guillermo Román Del Muro

INCO

217554557

Principios y prevención de defectos.

Computación Tolerante a fallas.

DR. MICHEL EMANUEL LOPEZ FRANCO.

Prevención de defectos:

“Más vale prevenir que curar”. En software estas expresiones se traducen en la observación común de que cuanto más tiempo permanezca en proceso un defecto, más costoso será solucionarlo. Analicemos algunas de las técnicas de prevención.

- Involucrar a los usuarios finales lo antes posible: Varios estudios han encontrado que la participación del usuario final es clave para la estabilidad de los requisitos y el éxito del proyecto del software.
- Crear un prototipo desechable: Cree un prototipo de interfaz de usuario desechable; presentar el prototipo a usuarios finales reales. Obtenga comentarios. Revisar el prototipo hasta que el usuario esté entusiasmado con el sistema. Luego construye el sistema. Este enfoque se correlaciona con una buena estabilidad de los requisitos y un bajo coste del sistema.
- Entregue el software de forma incremental en lugar de hacerlo todo a la vez: Escriba el código de producción para una pequeña del sistema. Ponga esta funcionalidad frente al usuario. Revise los requisitos, el diseño y el código hasta que el usuario esté entusiasmado con el sistema. Este enfoque no elimina por completo la dinámica de defecto-aumento de costos, para acortar el ciclo de retroalimentación desde los requisitos hasta la retroalimentación del usuario de una manera que reduce la cantidad de dependencias posteriores que se basarán en un trabajo erróneo.
- Realizar un taller de requisitos: Se ha descubierto que las técnicas rápidas de obtención de requisitos, como las sesiones JAD, son una forma eficaz de acortar el tiempo necesario para recopilar requisitos precisos y, al mismo tiempo, reducir la volatilidad de los requisitos en sentido descendente.
- Realizar análisis de casos de uso: En lugar de estar satisfecho con la primera explicación de los usuarios sobre lo que quieren que haga un sistema, examine los patrones de usos esperados del sistema para comprender mejor las necesidades reales del usuario.

- Primero cree el manual de usuario: La mayoría de los usuarios suelen entender mejor la explicación proveniente de un manual de usuarios a una especificación de requisitos tradicional.

Analicemos desde otra perspectiva otros métodos de prevención de defectos.

1. Análisis de requisitos de software: Es una de las principales causas del defecto en software. Tanto los requisitos como el diseño del software son importantes y deben de analizarse de manera eficiente y con mayor atención.
2. Revisión e inspección: Ambas son partes esenciales e integrales del desarrollo de software. Se consideran herramientas poderosas que pueden usarse para identificar y eliminar defectos si están presentes antes de que ocurran y afecten a la producción.
3. Registro y documentación de defectos: Se deben de mantener registros sobre los defectos para complementar simplemente la descripción del defecto. Este registro se puede utilizar aún más para comprender mejor los defectos.
4. Análisis de causa raíz: Es básicamente un análisis de la causa principal del defecto. Simplemente analiza qué provocó que ocurriera el defecto. Después de analizar la causa principal del defecto, se puede encontrar la mejor manera de evitar que se produzcan este tipo de defectos la próxima vez.

Bibliografía

- Steve McConnel. (2001). “An ounce of prevention”, IEEE software.
- GeeksforGeeks. (2020). Defect prevention methods and techniques. *GeeksforGeeks*. https://www.geeksforgeeks.org/defect-prevention-methods-and-techniques/