
UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

Materia: Computación Tolerante a Fallas

Nombre del alumno: Christian Daniel González García

Profesor: Michel Emanuel López Franco

Título de la actividad: Principio de prevención de defectos

Fecha: 05 de febrero 2024

La prevención de defectos es una parte fundamental en el proceso de desarrollo de software, ya que puede ayudar a mejorar la calidad del producto final y reducir los costos asociados con la corrección de defectos. Aquí tienes una lista de diferentes métodos para la prevención de defectos en el desarrollo de software:

## Revisión de código:

La revisión de código, también conocida como revisión de pares o inspección de código, implica que los desarrolladores revisen el código de sus compañeros en busca de errores, anomalías o malas prácticas. Esta técnica ayuda a detectar y corregir problemas antes de que se conviertan en defectos.

## Pruebas unitarias:

Las pruebas unitarias son pruebas automatizadas que verifican el comportamiento de unidades individuales de código, como funciones o métodos. Estas pruebas pueden ayudar a identificar y corregir errores en una etapa temprana del desarrollo.

## Análisis estático de código:

El análisis estático de código implica el uso de herramientas automáticas para analizar el código fuente en busca de posibles problemas, como errores de sintaxis, malas prácticas de programación o vulnerabilidades de seguridad.

## Gestión de la configuración:

Un sistema de gestión de la configuración ayuda a controlar y gestionar los cambios en el código fuente, la documentación y otros artefactos del proyecto. Un buen sistema de gestión de la configuración puede ayudar a prevenir errores causados por la introducción de cambios incorrectos o no autorizados.

## Estándares de codificación:

Establecer y hacer cumplir estándares de codificación consistentes puede ayudar a prevenir errores y mejorar la legibilidad y mantenibilidad del código. Los estándares pueden incluir convenciones de nomenclatura, estilo de codificación y directrices para el uso de comentarios.

## Pruebas de integración continua:

La integración continua implica la integración frecuente de cambios en el código base y la ejecución automática de pruebas para detectar problemas lo antes posible. Esto ayuda a prevenir la introducción de errores y garantiza que el código funcione correctamente en todo momento.

## Capacitación y educación:

Proporcionar capacitación y recursos educativos a los miembros del equipo sobre buenas prácticas de desarrollo de software, herramientas y técnicas puede ayudar a mejorar la calidad del código y prevenir la introducción de defectos.

## Revisión de requisitos:

Revisar y validar los requisitos del proyecto con los interesados y miembros del equipo puede ayudar a identificar y corregir problemas de comprensión, ambigüedad o inconsistencia en una etapa temprana del proyecto.

## Modelado y diseño:

Utilizar técnicas de modelado y diseño, como diagramas UML, prototipado y diseño arquitectónico, puede ayudar a prevenir errores al proporcionar una visión clara y detallada del sistema antes de su implementación.

## Gestión de riesgos:

Identificar y gestionar proactivamente los riesgos del proyecto puede ayudar a prevenir problemas y evitar la introducción de defectos graves durante el desarrollo del software.