UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Computación Tolerante a Fallas

Principios de prevención de defectos

Ingeniería en Computación

4 Septiembre 2023

Por: Carlos Uriel Salcido Aviña 217560751

Maestro: Michel Emanuel López Franco

Introducción

Es importante que, a la hora de programar, debamos tener en mente un plan para posibles errores, que pueden ocurrir en cualquier momento y por cualquier razón, como lo es en el caso de los errores casuados por hardware, como si se va la luz, hasta defectos causados por el usuario en el software, por si sale de una aplicación o programa sin guardar los cambios. Es aquí donde entran diversas formas para prevenir defectos y agilizar un programa.

Algunos métodos son los siguientes:

1. Redundancia: es la repetición de aquellos datos o hardware de carácter crítico que se quiere asegurar ante la posibilidad de fallos que pudieran surgir debido a diversas razones. Es una solución a los problemas de pretección y confiabilidad.
2. Monitorización y supervisión: Utilizar sistemas de monitorización para detectar signos tempranos de fallas y tomar medidas preventivas. Implementar alertas para notificar a los operadores sobre posibles problemas.
3. Virtualización: Utilizar tecnologías de virtualización y contenedores para aislar aplicaciones y servicios, de modo que las fallas en una parte del sistema no afecten al resto.
4. Copias de seguridad y recuperación: Realizar copias de seguridad regulares de datos críticos y establecer planes de recuperación de desastres para restaurar el sistema en caso de una falla catastrófica.
5. Actualizaciones y parches: Mantener el software y el hardware actualizados con las últimas correcciones de seguridad y actualizaciones de firmware para reducir la exposición a vulnerabilidades conocidas.
6. Pruebas y simulaciones: Realizar pruebas exhaustivas de sistemas y componentes para identificar y solucionar defectos antes de la implementación en producción. Utilizar herramientas de simulación para evaluar el comportamiento del sistema en condiciones de falla.

Se puede utilizar algoritmos y códigos de corrección de errores para detectar y corregir errores en datos o en la memoria. También diseñar software de manera que sea capaz de manejar errores inesperados y continuar funcionando de manera adecuada.

ODC (Orthogonal Defect Classification) es un framework sistemático para la clasificación de defectos de software, desarrollado por IBM en la década de 1990.

Utiliza información semántica de los defectos para extraer relaciones causa-efecto en el proceso de desarrollo.

Valores:

-Provee un rápido y efectivo *feedback* a los desarrolladores.

-Captura información de defectos que ocurrieron durante las fases del desarrollo y su uso en el área.

-Permite el entendimiento de qué errores son más frecuentes.

-Gracias a mediciones multi-dimensionales y el análisis que se otorga, los desarrolladores pueden manejar propiamente el proceso de desarrollo y la caliad del producto.

Un defecto pasa por dos secciones con ODC. *Opener Section* es cuando se encuentra un defecto, y se les puede clasificar los siguientes atributos:

-*Activity*: es la actividad que se está realizando mientras sucede y se descubre el error.

-*Trigger*: Las condiciones que se tuvieron que dar para que surgiera el defecto. ¿Qué se necesita para reproducir el defecto? Mencionar la descripción del ambiente o condición que actúa como catalizador del error.

-*Impact*: Mencionar el impacto que crees que el defecto tendría en el cliente, o el que ya tuvo en el cliente.

La otra sección se conoce como *Closer Section*, que sucede cuando sabes cómo el defecto fue arreglado, se les puede clasificar los siguientes atributos:

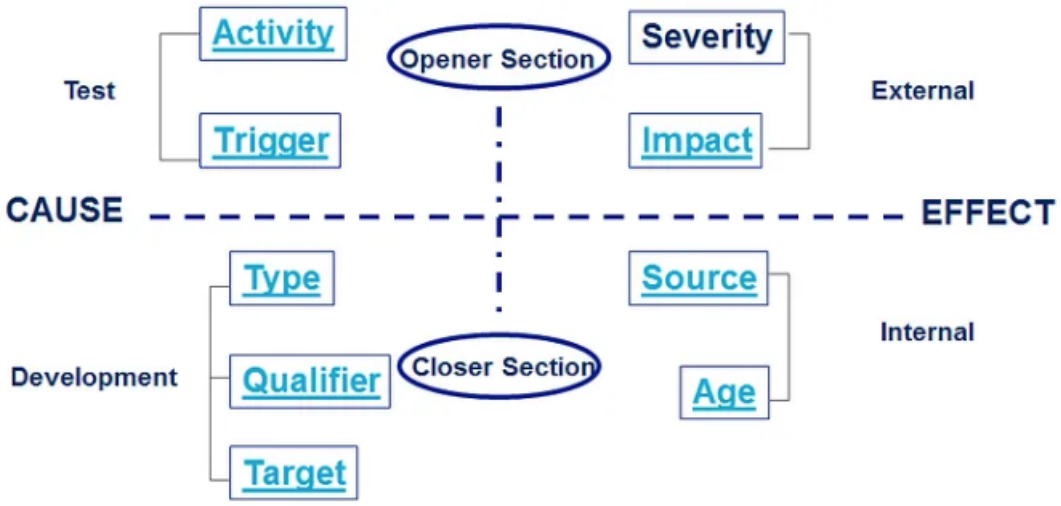
-*Target*: Representa la identidad de alto nivel de la entidad que se arregló.

-*Defect Type*: Representa la naturaleza de la corrección que se hizo.

-*Qualifier*: Aplica al *Defect Type*, captura el elemento de una implementación inexistente, con error o si es irrelevante.

-*Source*: Identifica el origen del *Target* que tenía el defecto.

-*Age*: Identifica la historia del *Target* que tenía el defecto.



Conclusión

Me parece interesante las formas en que un equipo de personas puede cuantificar los errores, y cómo de hecho vale la pena invertir en clasificarlos y analizarlos todos como un conjunto, según se pudo leer en el pdf que nos compartió el profesor, para así poder identificar factores comunes y saber qué medidas tomar para disminuir la cantidad de errores y defectos que se cometen durante el desarrollo de un proyecto, ahorrando así tiempo y dinero.

Referencias

Vasueda V (2016) “What is Orthogonal Defect Classification (ODC)? by Vivek Vasudeva”, en medium. Disponible en: https://medium.com/@SWQuality3/what-is-orthogonal-defect-classification-odc-by-vivek-vasudeva-f2e49917f478#:~:text=ODC%20is%20a%20concept%20that,Escape%20and%20Root%20Cause%20Analysis.

Khadir A (2013) “Software Excellence Augmentation through Defect Analysis and Avoidance”, en STAR Journal.