INSTITUTO TECNOLÓGICO SUPERIOR ZACATECAS OCCIDENTE

MATERIA: VALIDACIÓN Y VERIFICACIÓN DE SOFTWARE

CARRERA: INGENIERIA EN SISTEMAS COMPUTACIONALES

TEMA: INVESTIGACIÓN Y MAPA

DOCENTE:
I.S.C. ERICKA JAZMÍN ROBLES GÓMEZ
ELABORADO POR:
FABIAN ARMANDO HERRERA AVALOS
OSIEL BARRIENTOS RAMIREZ
MIRANDA ARROLLO JARA
SORAYA CASTILLO MENA

FECHA: 29 DE ENERO DEL 2016

INTRODUCCIÓN

Cuando hablamos de errores uno se imagina cosas que están mal hechas, que salieron mal, o que simplemente no dan los resultados esperados. Pues en parte podríamos decir que estamos medio acertados en nuestra respuesta, mas sin embargo un error lleva más factores para ser considerado en si eso, además, ¿cuál es la diferencia entre falla y error?, o ¿error y defecto?, la verdad, para saber exactamente nuestra respuesta, sería necesario ver el significado de cada uno, asi como algunos ejemplos.

A continuación veremos las definiciones de cada uno de los puntos, ejemplos y algunos tipos de los mismos.

¿Qué es un error?

Es una equivocación cometida por el desarrollador. Algunos ejemplos de errores son: un error de digitación, una malinterpretación de un requerimiento o de la funcionalidad de un método. El estándar 829 de la IEEE coincide con la definición de diccionario de error como "una idea falsa o equivocada". Por tal razón un programa no puede tener o estar en un error, ya que los programas no tienen ideas; las ideas las tienen la gente.

¿Qué es un defecto?

Un defecto se encuentra en un artefacto y puede definirse como una diferencia entre la versión correcta del artefacto y una versión incorrecta. Coincide con la definición de diccionario, "imperfección".

¿Qué es una falla?

En terminología IEEE (Institute of Electrical and Electronics Engineers), una falla es la discrepancia visible que se produce al ejecutar un programa con un defecto, el cual es incapaz de funcionar correctamente (no sigue su curso normal).

Tipos de errores y defectos

Las tres grandes clases clasificaciones de errores del software son los de requisitos, de diseño y de instrumentación.

Errores de diseño

Se introducen por fallas al traducir los requisitos en estructuras de solución correctas y completas, por inconsistencias tanto dentro de las especificaciones de diseño y como entre las especificaciones de diseño y los requisitos. Un error de requisitos o un error de diseño, que no se descubre sino hasta las pruebas de código fuente, puede ser muy costoso de corregir. De modo que es importante que la calidad de los requisitos y de los documentos del diseño se valoren pronto y con frecuencia.

Errores de instrumentación

Son los cometidos al traducir las especificaciones de diseño en código fuente. Estos errores pueden ocurrir en las declaraciones de datos, en las referencias a los datos, en la lógica del flujo de control,

en expresiones computacionales, en interfaces entre subprogramas y en operaciones de entrada/salida.

Ejemplos

Defectos de diseño de programas

- Diseños con colores inapropiados para las personas que padecen daltonismo
- Diseños que usan textos con tipografías de difícil lectura por su tamaño o diseño
- Diseños que fuerzan el uso del ratón o mouse sin dejar alternativas de teclado para personas con disfunciones motrices
- Diseños con implicaciones culturales, por ejemplo usando partes del cuerpo que en una determinada cultura sean objeto de vergüenza o burla o símbolos con características de identidad cultural o religiosa
- Estimar que el equipo donde se instalará tiene determinadas características (como la resolución de la pantalla, la velocidad del procesador, la cantidad de memoria o conectividad a internet) propias de un equipo de gama alta, en vez de diseñar el software para su ejecución en equipos normales

Errores de programación comunes

- División por cero
- Ciclo infinito
- Problemas aritméticos como desbordamientos (overflow) o subdesbordamientos (underflow).
- Exceder el tamaño del array
- Utilizar una variable no inicializada
- Acceder a memoria no permitida (Violación de acceso)
- Pérdida de memoria (memory leak)
- Desbordamiento o subdesbordamiento de la pila (estructura de datos)
- Desbordamiento de búfer (buffer overflow)
- Bloqueo mutuo (deadlock)
- Indizado inadecuado de tablas en bases de datos.
- Desbordamiento de la pila de recursión, cuando se dejan demasiadas llamadas en espera.

Defectos de instalación o programación

- Eliminación o sustitución de bibliotecas comunes a más de un programa o del sistema (DLC Hell).
- Reiniciar arbitrariamente la sesión de un usuario para que la instalación tenga efecto.
- Suponer que el usuario tiene una conexión permanente a internet.
- Utilizar como fuente enlaces simbólicos a ficheros que pueden cambiar de ubicación.

Conclusiones

Es imposible garantizar que un producto no tiene Defectos, sin embargo, es posible mejorar su proceso para producir productos con muy pocos defectos, con una tasa de defectos controlada.

La estrategia es identificar las métricas que podrían indicar el posible número de defectos restantes en el producto final y establecer un estándar que pueda ser conocido por nuestros clientes, según el cual nuestro producto tiene un X número de defectos que pueden llegar a convertirse en Fallas del sistema.

Si identificamos cada una de las posibles fallas a tiempo, nuestro sistema será un producto de calidad con mínimos posibles defectos.