GLOSARIO

Causa Raíz: se refiere al origen o la fuente fundamental de un defecto o problema en el software. Identificar la causa raíz es crucial para comprender por qué ocurrió un defecto y cómo se puede prevenir en el futuro. Encontrar la causa raíz implica analizar en profundidad las circunstancias, condiciones, acciones o decisiones que condujeron al defecto. Esto puede implicar revisar el código, los procesos de desarrollo, las especificaciones, entre otros aspectos del proyecto.

Defecto: Es una discrepancia o anomalía en el software que puede causar un error. Es una imperfección en el producto, como un error en el código o una funcionalidad que no cumple con los requisitos establecidos. Los defectos son el resultado de errores en el proceso de desarrollo o en la implementación del software.

Error: Es una acción humana que produce un resultado incorrecto o inesperado en el software. Puede ser el resultado de un malentendido de los requisitos, un error de codificación, un mal diseño, entre otras causas. Los errores también pueden denominarse "bugs".

Fallo: Ocurre cuando el software no realiza la función que se espera que realice. Es el resultado observable de un defecto en el software que provoca un comportamiento incorrecto o inesperado durante la ejecución. Un fallo puede ser detectado durante las pruebas o cuando el software está en producción y en uso por los usuarios finales.

Principio de Pareto: El principio de Pareto, también conocido como la regla del 80/20, es un concepto que sugiere que, en muchas situaciones, aproximadamente el 80% de los efectos provienen del 20% de las causas. Este principio se basa en observaciones del economista italiano Vilfredo Pareto, quien notó que aproximadamente el 80% de la tierra en Italia estaba en manos del 20% de la población. Es una herramienta útil para identificar y priorizar los elementos más importantes o significativos en una variedad de contextos, lo que puede ayudar a optimizar recursos, esfuerzos y resultados.

Probar: Proceso esencial para evaluar y mejorar la calidad del software, reduciendo el riesgo de fallos en su funcionamiento. Consiste en descubrir defectos y evaluar la

calidad de los artefactos de software, llamados objetos de prueba. Incluye tanto pruebas dinámicas (ejecución de software) como estáticas (revisiones y análisis), y no se limita solo a verificar si el software cumple con los requisitos, sino que también valida si satisface las necesidades de los usuarios. Requiere planificación, gestión y control adecuados, así como habilidades analíticas por parte de los probadores.

Pruebas Estáticas: Son pruebas que se realizan sin ejecutar el código del programa. Se centran en revisar los artefactos de software, como el código fuente, especificaciones, diseños, documentos y otros elementos, para identificar posibles defectos. Las pruebas estáticas incluyen técnicas como revisiones de código, análisis estático y walkthroughs, y son útiles para detectar problemas tempranamente en el proceso de desarrollo.

Pruebas Dinámicas: Son pruebas que se llevan a cabo ejecutando el código del programa y observando su comportamiento en tiempo de ejecución. Estas pruebas evalúan el funcionamiento real del software y su cumplimiento con los requisitos especificados. Las pruebas dinámicas incluyen la ejecución de casos de prueba y técnicas como pruebas de unidad, integración, sistema y aceptación. Son esenciales para verificar el comportamiento funcional y detectar posibles errores durante la ejecución del software.

SDLC: (Ciclo de Vida del Desarrollo de Software, por sus siglas en inglés) es un marco o metodología utilizada por los equipos de desarrollo de software para planificar, diseñar, crear, probar e implementar un sistema o aplicación de software. Es un proceso estructurado que guía el desarrollo de software desde la concepción de la idea hasta su entrega y mantenimiento. El SDLC consta típicamente de varias fases, que pueden variar según el modelo específico de desarrollo utilizado, pero comúnmente incluyen: requisitos, diseño, implementación, pruebas, despliegue, mantenimiento. El SDLC proporciona una estructura organizada para el desarrollo de software, lo que ayuda a garantizar la calidad, la consistencia y la eficiencia en todo el proceso. Sin embargo, existen varios modelos de SDLC, como el modelo en cascada, el modelo en espiral, el modelo iterativo e incremental, entre otros, cada uno con sus propias características y enfoques específicos.

Técnica de Prueba: Métodos sistemáticos y estructurados utilizados para evaluar la calidad y el funcionamiento de un software. Estas técnicas se aplican durante el proceso de desarrollo y pruebas de software para detectar defectos, validar la funcionalidad y verificar si el software cumple con los requisitos especificados. Existen muchas técnicas de prueba disponibles, y la elección de la técnica adecuada

depende de factores como los requisitos del proyecto, el tipo de software y los objetivos de prueba específicos.

Validación: Es el proceso de evaluar si el sistema cumple con las necesidades y expectativas del cliente o usuario final. Se trata de asegurar que el software que se está construyendo es el producto correcto. La validación implica comprobar si el software cumple con los requisitos y expectativas del usuario en el mundo real. Esto puede incluir pruebas de aceptación del usuario, pruebas beta, entre otras actividades.

Verificación: Es el proceso de evaluar si un sistema o componente cumple con las especificaciones y requisitos establecidos durante el desarrollo. Esencialmente, se trata de asegurar que el software se está construyendo correctamente. La verificación se enfoca en la revisión de documentos, modelos y código para detectar errores en el proceso de desarrollo.