**Test Quality**

**Prework**

* Definir el concepto de calidad en software.
* Diferenciar entre testing y calidad.
* Crear pruebas de calidad.
* Utilizar herramientas y métricas que ayuden a definir la calidad.

**Aspectos de la calidad del software:**

* **Buen diseño:** Siempre es importante tener un diseño bueno y estético para complacer a los usuarios.
* **Fiabilidad:** aunque se trate de un software, debe ser capaz de realizar sus funciones de forma impecable y sin problemas.
* **Durabilidad:** La durabilidad es un término confuso, en este contexto, la durabilidad significa la capacidad del software para trabajar sin ningún problema durante un largo período de tiempo.
* **Consistencia:** El software debe ser capaz de funcionar de forma consistente en todas las plataformas y dispositivos.
* **Mantenibilidad:** Los errores asociados a cualquier software deben poder capturarse y solucionarse rápidamente y las nuevas tareas y mejoras deben añadirse sin ningún problema.
* **Relación calidad-precio:** El cliente y las empresas que hacen esta aplicación deben sentir que el dinero gastado en esta aplicación no se ha desperdiciado.

**Modelos de calidad de software**

* **Modelo de Calidad de McCall.**

Objetivo: mantener la armonía entre usuarios y desarrolladores.

* **Modelo de Calidad de Boehm.**

Nos ayuda a medir la calidad del software en función de determinadas características.

* **Modelo de Calidad de Dromey.**

Se centra en los atributos y subatributos para conectar las propiedades del software con los atributos de calidad.

Elementos principales:

* + Propiedades del producto que afectan a la calidad.
  + Atributos de calidad de alto nivel.
  + Vinculación de las propiedades con los atributos de calidad.

Las **pruebas de calidad** incluyen todas las actividades que se centran en la aplicación de normas y procedimientos asociados a la garantía de que el software cumple con un determinado conjunto de requisitos antes de su lanzamiento al público.

Las organizaciones deben definir primero un conjunto medible de métricas de calidad y un proceso para garantizar que esas normas estén representadas en el software. Lo que significa que las organizaciones necesitan obtener una comprensión completa de lo que el usuario final siente como una "experiencia de calidad".

Además de gestionar y eliminar los errores, es vital determinar qué **atributos de calidad** debe haber.

**Gestión de Calidad**

* **Estándar de codificación:** Una de las mejores maneras de ofrecer un producto de alta calidad es utilizar el estándar de codificación. Garantiza que cada miembro del equipo está haciendo bien su trabajo y conduce a la consistencia y legibilidad del código. Tener un estándar facilita el uso del proyecto y mejora la calidad del software.
* **Analizar el código:** Los especialistas con experiencia saben que es más fácil prevenir los problemas que tratarlos después de la publicación. La calidad debe ser la prioridad número uno durante todo el proceso de desarrollo. Cuanto antes se determinen los errores, más rápido, fácil y barato será solucionarlos. Por eso los desarrolladores profesionales analizan el código de inmediato.
* **Utilizar las últimas tecnologías:** Es mejor no confiar sólo en los desarrolladores, sino también utilizar las métricas mencionadas anteriormente. Una comprobación manual del código sigue siendo útil, pero no es tan eficaz. Deja que las métricas de calidad del desarrollo de software sean automatizadas.
* **Refactorización**: Si se trata de mejorar un producto ya existente y obsoleto, utiliza la refactorización. Ayuda a limpiar la base de código y hace que sea mucho más fácil de usar. Lo mejor es hacerlo gradualmente.

**Tareas que garantizan calidad:**

* Definir las características que definen la calidad de un producto.
* Decidir cómo medir cada una de esas características de calidad.
* Establecer normas para cada característica de calidad.
* Hacer un control de calidad con respecto a las normas.
* Averiguar las razones que obstaculizan la calidad.
* Realizar las mejoras necesarias.

**Métricas de control de calidad en las pruebas de software nos ayudan a:**

* Identificar los principales problemas que hay que probar.
* Elegir un especialista que sepa qué hacer con las métricas.
* Probar los aspectos más importantes del software.
* Calcular la eficacia de la métrica.
* Observar si necesita algún cambio.
* Mejorar los aspectos que desea cambiar.

**Métricas ágiles.**

Una métrica ágil es útil cuando necesitas mejorar el proceso de desarrollo. Tiene en cuenta el tiempo de espera y el ciclo, la velocidad, así como el porcentaje de apertura y cierre.

* **Tiempo de espera:** Es el tiempo que los ingenieros emplean en idear, diseñar, desarrollar y terminar el proyecto de software. Si se acorta el tiempo, se puede lanzar el producto más rápidamente y llamar la atención del consumidor. Como no se les hará esperar mucho tiempo, su fidelidad aumentará.
* **Tiempo de ciclo:** Puede ser difícil entender la diferencia entre estas dos definiciones, pero no son lo mismo. El periodo de ciclo comienza con el desarrollo de la aplicación y termina cuando se ha completado, mientras que el plazo de entrega comienza con la recepción del pedido y termina con su entrega.
* **Velocidad:** Calcula el tiempo que necesitarán los programadores para desarrollar un producto. Ayuda a entender cuánto tiempo necesita el equipo para cada etapa. Así, se puede hacer un plan para futuros productos según los análisis ya existentes.

**Métrica de producción.**

Esta métrica estima la cantidad de trabajo que los desarrolladores ya han realizado, su productividad y velocidad. Se puede comprobar mediante los días activos, el tiempo de fallos y reparaciones, la productividad, los alcances de las tareas y otros factores.

* **Días activos:** Es el tiempo que los desarrolladores dedican a la codificación. No incluye ningún otro tipo de actividades menores, como la planificación. Esta métrica ayuda a identificar los costes ocultos.
* **Tiempo de fallos y reparaciones:** Cuando se desarrolla un producto desde cero, nunca se pueden evitar los errores y los fallos. Por eso, lo único que se puede hacer es anotar el tiempo que los ingenieros dedican a resolver el problema.
* **Productividad:** Es difícil medir con precisión este aspecto, pero el volumen de código de cada desarrollador puede servir de referencia.
* **Alcance de las tareas:** Es el volumen de código que un desarrollador puede producir al año. Parece raro, pero ayuda a calcular cuántos ingenieros se necesitarán para un proyecto.
* **Volumen de código:** Es el volumen de código que se ha modificado en el producto.

**Métricas de respuestas de seguridad**

Garantizan la seguridad del producto. Cuando se mide la calidad del software, hay que comprobar cómo responde la aplicación a la seguridad. Es importante comprobar la rapidez con la que tu proyecto puede detectar un problema y eliminarlo, o al menos alarmar al responsable de TI al respecto.

**Antigüedad de las dependencias**

Debes asegurarte de que todas las dependencias de tu base funcionan correctamente. Es posible que algunas de ellas deban ser actualizadas.

**Medidas orientadas al tamaño**

Calcula el tamaño del código y determina los fallos, errores y costes por cada 1000 líneas. Ayuda a medir la calidad de la aplicación en función de su tamaño y de la precisión del código.

**Métodos orientados a la función**

Muestra la cantidad de funcionalidad empresarial que se puede obtener del producto. Es el principal cuantificador y analiza toda la información disponible, como las entradas y peticiones de los usuarios, los informes, los mensajes sobre los errores y las peticiones de los usuarios.

**Métrica de defectos**

La cantidad de defectos es el principal indicador de la calidad del software. Incluye:

* Las etapas en las que surgen los defectos.
* El número de informes de defectos.
* El tiempo necesario para identificar y eliminar los defectos.
* El número de defectos por línea de código (densidad).

**Pull Requests**

La complejidad del proyecto, el compromiso de los pull requests y la interacción del equipo:

* Pull requests que no han pasado el conjunto de pruebas.
* Pull requests que rompieron la compilación.
* El número de solicitudes rechazadas y fusionadas.
* El número de comentarios de los pull requests.

**Técnicas de Revisión de la Calidad**

Una revisión es una forma de utilizar la diversidad de un grupo para lo siguiente:

* Resaltar las mejoras necesarias en el producto que elaboró una sola persona o equipo;
* Confirmar aquellas partes de un producto en las que no se desea o no se necesita hacer una mejora;
* Realizar el trabajo técnico de calidad más uniforme, o al menos más predecible, que pueda lograrse sin hacer revisiones, a fin de que el trabajo técnico sea más manejable.

**Efecto de los defectos del software en el costo**

* **Defecto y Falla:** los dos implican un problema de calidad.
* **Error:** denota un problema de calidad descubierto por ingenieros de software antes de entregar el software al usuario final.
* **Objetivo:** encontrar errores durante el proceso a fin de que no se conviertan en defectos después de liberar el software.
* **Beneficio**: descubrimiento temprano de los errores, de modo que no se propaguen a la siguiente etapa del proceso del software.

**Revisiones Informales**

No hay una planeación o preparación por adelantado, ni agenda o estructura de la reunión, y no se da seguimiento a los errores descubiertos, la eficacia de tales revisiones es mucho menor que la de los enfoques más formales.

**Revisiones Formales**

Su objetivo es:

* Descubrir los errores en funcionamiento, lógica o implementación.
* Verificar que el software que se revisa cumple sus requerimientos.
* Garantizar que el software está representado de acuerdo con estándares predefinidos.
* Obtener software desarrollado de manera uniforme.
* Hacer proyectos más manejables.

**Reunión de Revisión.**

* Deben involucrarse de tres a cinco personas.
* Preparación previa, pero no más de dos horas de trabajo.
* Debe ser de al menos dos horas.
* Centra en una parte específica (y pequeña) del software general.

**Reporte y Registro de la Revisión.**

* Se produce la lista de pendientes de la revisión.
* Elabora un reporte técnico formal de la revisión.
* El reporte de la revisión es una sola página (quizá con anexos).
* Identificar las áreas de problemas en el producto.

**Lineamientos para la Revisión**

* Revisar el producto, no al productor.
* Establecer una agenda y seguirla.
* Limitar el debate y las contestaciones.
* Enunciar áreas de problemas, pero no intentar resolver cada uno.
* Tomar notas por escrito.
* Limitar el número de participantes e insistir en la preparación previa.
* Desarrollar una lista de verificación para cada producto que sea probable que se revise.
* Asignar recursos y programar tiempo para las RTF.
* Dar una capacitación significativa a todos los revisores.
* Revisar las primeras revisiones.

**Calidad de software**

Son características propias del software, aquellas que tu quieres controlar y asegurar.

**Work**

* Definir el concepto de calidad en software
* Diferenciar entre testing y calidad
* Crear pruebas de calidad
* Utilizar herramienta y métricas que ayuden a definir la calidad

**Postwork**

* Desarrollar funcionalidad adicional a nuestro programa.
* Calcular métrica de complejidad ciclomática