Testing de Software (Unidad 4: Aseguramiento de calidad del proceso y del producto)

Testing testea código, es decir, relacionado al producto.

Esquema de temas

TDD Tiene que ver con desarrollo, no de testeo. Para pruebas unitarias.

Contexto de testing:

Diferencia entre testing y aseguramiento de calidad: QA apunta a prevención/aseguramiento, las cosas que se hacen MIENTRAS se desarrolla el software para asegurar su calidad; testing se realiza cuando el producto ya está construido, entonces se hace CONTROL de calidad, no ASEGURAMIENTO de calidad. Todo forma parte de la disciplina de aseguramiento de calidad del software, pero no es exactamente lo mismo. Si solo haces testing, no haces QA de manera global.

El testing es **un proceso destructivo, en el cual se intenta encontrar defectos, cuya presencia se asume, en el código. Testing exitoso es aquel que encuentra los defectos. Si no se encuentra defectos, significa que es un mal testing porque se asume que defectos hay siempre, debido a el error humano siempre presente.**

No es ético, no es profesional no realizar testing. Se utiliza 30% de las horas de desarrollo, no de proyecto. Se incluye el tiempo de corrección de desarrollo. Las pruebas unitarias son parte del tiempo de desarrollo. El testing es lo que más consume o lo que más debería consumir.

La calidad se inserta durante el proceso de creación/desarrollo, no al final del proceso mediante controles.

El testing llega tarde, controla para validación y verificación.

Error no reproducible no es error. Testing ad hock o algo así.

Tendencias a testing automatizado.

Edad del defecto: Tiempo que transcurre entre que inserta el defecto y se detecta. A mayor edad, mayor costo para corrección.

Los errores pueden surgir en cualquier parte, por lo que el testeo sigue siendo necesario, aunque se trabaje con prevención.

XP utiliza modelado y programación de a pares a modo de prevención. Detectar el error en el momento que se genera, es costo 0.

Error vs Defecto vs Falla: Error es cuando no se traslada desde la etapa en la que se produjo. Testing encuentra defectos, porque son de una etapa anterior a la que se encuentran actualmente. La falla es cuando llevan a malfuncionamiento a partir de un defecto en el software. Algunos defectos generan fallas, otros no.

Categorización de defectos:

- **Severidad: Indica daño que realiza en el software. Asignado por quien realiza el testing. Línea entre crítico y mayor es muy sutil.**
    - **Bloqueante: Impide el uso total**
    - **Crítico: Impide una función en concreta, pero a partir de esa funcionalidad concreta, no todo el sistema**
    - **Mayor: Defecto que genera una falla pero que tal vez no impide el trabajo normal, como mucho hay advertencias, pero se puede seguir usando.**
    - **Menor: Errores de advertencia, podes seguir trabajando, pero no se cambia una funcionalidad.**
    - **Cosmético: Solo algo visual, no impide funcionalidad**
- **Prioridad: Ligado no solo a severidad, si no a importancia del negocio.**
    - **Urgencia**
    - **Alta**
    - **Media**
    - **Baja**

Niveles de prueba: El software se desarrolla de general a particular, y se prueba de particular a general.

- **Pruebas unitarias:** Ver si lo construido por cada desarrollador funciona, o no. Va en contra de los principios de que el desarrollador no debería probar su propio código. Se utiliza TDD. Sigue siendo parte de implementación.
- **Pruebas de integración:** También se conoce como prueba de interfaces. Se asume que cada componente de manera aislada funciona bien, y se hace foco en comunicación entre componentes. Se puede integrar y funcionar como unidad. Sigue siendo parte de implementación. Se arma bin y es lo que se pasa a testing, para ir a pruebas de sistema. Esto del bin no lo entendí muy bien no sé ni si se escribe así, ojito con eso, investigar
- **Pruebas de sistema:** Se testea lo que se tiene hasta el momento. Testing funcional, manual, a nivel de funcionalidades del negocio. Se deberían probar requerimientos no funcionales, difíciles de probar debido al entorno. La automatización permite simular situaciones.
- **Pruebas de aceptación de usuario:** UAT user acceptance test. El usuario debe estar presente. Determina si te pagan por el trabajo o no. Aceptación (o no) por parte del cliente.

Ambientes para construcción de software

- **Desarrollo**
- **Prueba: Sistema**
- **Pre-Producción: Usuario**
- **Producción**

Tomar bases de datos de producción y se los despersonaliza, para tener datos reales pero no datos sensibles. Trabajar con datos que cumplan características, y para cumplir performance.

Un caso de prueba es un artefacto producido por el testeo. En US, son títulos.
Los casos de prueba DESCRIBEN el paso a paso de una situación concreta ejemplificadora de UNA función del sistema, con todos los datos para ejecutar la prueba más los datos de contexto (Situación del sistema para realizar la prueba: Info de base de datos, datos a cargar) bien establecido en diseño de caso de pruebas.

El acceso a datos mediante roles es algo que se paga mucho por alto durante el testing. Ya sea la falta de permisos como el exceso de permisos. El sistema no te debería permitir hacer x cosa, y funciona bien la prueba si falla.

Testing ad hoc: Da lugar a defectos no reproducibles.

Los casos de prueba son importantes para poder reproducir el error, al tener los datos de seteo/precondiciones para efectivamente reproducir el defecto y corregirlo rápidamente.

Derivación de casos de prueba: Desde dónde se saca info para realizar las verificaciones. Problemas si las US no tienen criterios de aceptación.

- **El esquema de PP va de abajo hacia arriba, los de abajo son muy abstractos y los de arriba son demasiado técnicos, el punto medio sería el caso ideal.**

Condiciones de prueba: Entorno sobre el cual se ejecutan las pruebas.

Los casos de prueba tienen ejercicios en la UV para el miércoles que viene, leer!!!!

Objetivo:
Datos de entrada y de ambiente:
Comportamiento esperado:

Para cada escenario de un caso de uso, se realiza un caso de prueba con datos específicos.

Estrategias de testing:

- **Caja blanca: Estático. Es ver el código en busca de algo, sin ejecutar el sistema.**

- **Caja negra: Dinámico, se ejecuta el sistema.**

Métodos de implementación para estrategias de testing

- **Caja negra**
    - **Basado en especificaciones**
        - **Partición de equivalencias**
        - **Análisis de valores límites:** Si se quiere probar una fecha, se toma el valor entero dependiendo del día, mes y año. Se prueban rangos. Se prueban números externos al intervalo válido, y números dentro del intervalo válido, para ver si funcionan. Se prueban solamente los del valor límite del rango (Si es de 1 a 31, se prueba 0 y 32, y 1 y 31. Los intermedios deberían ser válidos y los otros no).
        - **Reducen cantidad de casos de prueba a diseñar, ayudan a determinar cuántos se deberían diseñar, sin reducir su efectividad ni calidad. Menor cantidad de casos de prueba para detectar mayor cantidad de errores**
    - **Basados en la experiencia**
        - **Adivinanza de defectos**
        - **Testing Exploratorio:** Es para ganar experiencia mediante testing genérico.
- **Caja blanca: Métodos de cobertura. Establecen cuánto código se va a cubrir.**
    - **Cobertura de enunciados o caminos básicos:** Se prueba al menos una vez cada camino básico.
    - **Cobertura de sentencias**
    - **Cobertura de decisión**
    - **Cobertura de condición**
    - **Cobertura de decisión/condición**
    - **Cobertura múltiple:** La que más cumple y que te va a asegurar que todas las líneas de código se van a probar al menos una vez.

Concepto de ciclo de test/prueba: Periodo de tiempo en el que un tester recibe una versión de un producto, ejecuta todos los casos de pruebas planificados. Desde inicio hasta el fin.

El primer ciclo se llama ciclo cero. Parametrización, es manual.
La próxima versión es el ciclo 1, 2, etc…
La situación ideal es que sea ciclo 0, y ciclo 1. Máximo 3 sería el plan.

Concepto de regresión: Cuando se prueba sin regresión, los casos de prueba son binarios: Pasan o fallan. Al volver, solo se prueban los que fallaron en el ciclo anterior, eso sería sin regresión. El problema es que si toca algo de la lógica se puede romper

otro de los casos de prueba, entonces la regresión sería si se vuelven a probar todos los casos de prueba de vuelta.

El testing pone en evidencia algunos defectos, solo los encontrados, hay otros que NO se ven.