ISW
07-10-25 / martes / meles

TESTING
|—> se lo define como actividad destructiva. Su objetivo es encontrar defectos cuya presencia se asume en el sw.

En el contexto del PUD está en la prueba (diagrama rosa de la clase pasada)

Requerimiento – Análisis – Diseño – Implementación – Prueba – Despliegue

-->

Proceso–> Planificacion – *Plan de prueba*

–> Diseño – ***Casos de prueba***

–> Ejecucion – *Reporte de defecto*

–> Cierre –

-->

Actividades de la prueba: – Validar

– Verificar

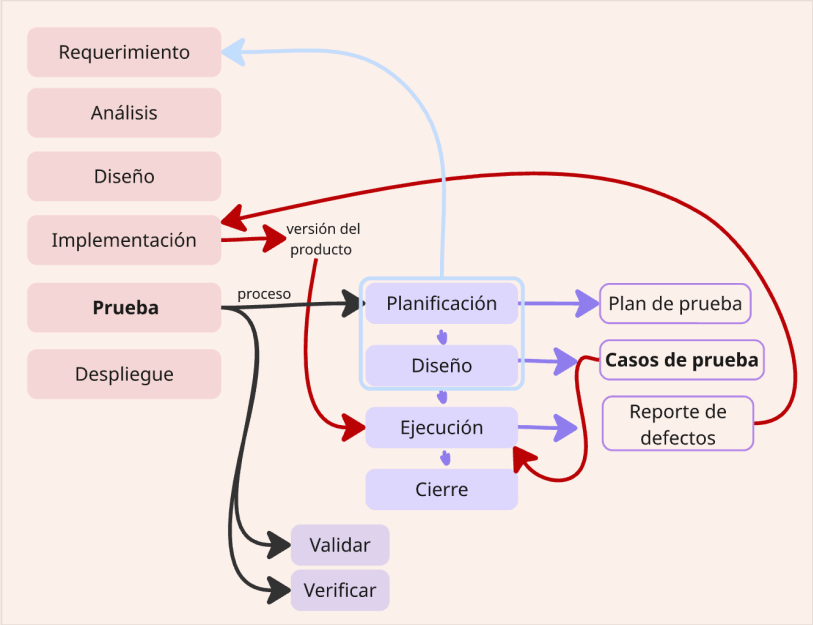
(el artefacto que sale de cada proceso de la prueba es el que tienen al lado)
Casos de prueba es el artefacto fundamental para el testing Se definen a partir de requerimientos

Cuando de la implementación sale una versión del producto, va a la etapa de ejecución con los ***casos de prueba***, y podemos obtener los defectos que aparecen

- *Reporte de defectos* es una lista de cosas que se encuentran en el producto. Se le devuelve a la implementación, porque los deben corregir los desarrolladores.
  - Se vuelve a enviar a ejecución con otra versión

Lo que yo puedo adelantar del testing es la planificación y el diseño. Acorta el tiempo.

Que, Como, Cuando, Quien - Estas preguntas se hacen en el plan de testing
Sin casos de prueba, no se puede testear (caso sistemático)



Conceptos importantes

- *Testing Metodológico vs AdHoc*
  - TM es el que se hace siguiendo un proceso. Es el que diseña casos de prueba, y los usa.
  - AdHoc es el testing que no se puede reproducir (no es útil para el dev que tiene que corregir). No metodológico, no sistemático. No tiene estructura.

Testing exitoso es el que encuentra error
"Testing que no se puede reproducir, no es error"

- Validar ≠ Verificar
  - Validación: proceso que tiene que ver con adecuación. Vemos si hace lo que realmente tiene que hacer
  - Verificación: ve si lo que tiene que hacer lo hace bien

En el testing tengo que hacer las dos cosas. (validar y verificar)

Los errores de validación son los más caros de corregir. (el origen del error es requerimiento)
El error de programación es corregible barato

Conceptos importantes

- *Error vs Defecto*
  - Error: falla en el sistema (como consecuencia de nosotros). No se trasladó. Baratos
  - Defecto: es un error que se traslada de una etapa anterior a una posterior. Si provoca un mal funcionamiento en el sistema, es una falla

Va a haber defectos porque el sw es una tarea humano intensiva !!

No es posible el testing exhaustivo, tiene un costo que no se justifica

- EDAD DEL DEFECTO: se mide como el tiempo desde que se introdujo, hasta que se corrigió
  el crecimiento del costo del defecto es exponencial. El costo depende de la edad del defecto.

Conceptos importantes

- *Severidad del defecto:* no tiene que ver con lo que me cuesta corregirlo, sino con el impacto que provoca en el uso del sistema. Los grados los define el equipo de testing.

    - *Bloqueante/invalidante: no se puede usar el sistema*
    - *Grave:*
    - *Leve:*
    - *Cosmética.* ortografia, de interacciones de usuario, de formato (por ej el cuit, a-aaaaa-a)
    - ***mejora** (no es un defecto, pero es algo que suelen usar los de testing para indicar que se puede mejorar)*
- *Prioridad*: lo asigna el cliente
    - *Alta*
    - *Media*
    - *Baja*

NIVELES DE TESTING

Se lo prueba de lo más chiquito a lo más grande (al revés de como se desarrolla, que se empieza de lo más general a lo más especifico)

- **De aceptación de usuario**:
- **de sistema o version**:
- **Integración / de interfaces**: el resultado es un build, una versión del producto que va al nivel de arriba
- **Unitaria**: (la que usa tdd). Primer nivel de testing, el más bajo.

Según los libros, integración y sistemas se hace en prueba. Pero no es así siempre por el tiempo de los testadores.

Prueba de aceptación de usuario	En despliegue
Prueba de sistema o versión	SI o si en prueba
Prueba de integración o de interfaces	Prueba o implementación
Prueba unitaria	Implementación

Principio de testing: "Una unidad de desarrollo no debería probar su propio desarrollo"

- Testing es una situación de control.

AMBIENTES DE SW

- **Desarrollo**: Pruebas untarias/ de integración
- **Prueba**: Pruebas de sistema
- Pre-Producción: pruebas de aceptación de usuario
- **Producción**:

Como mínima, los en negrita. Lo ideal es que también esté pro-producción (las pruebas de aceptación de usuario)
ciclo de prueba: abarca la ejecución de la totalidad de los casos de prueba establecidos aplicados a una misma versión del sistema a probar.

- ciclo cero: el primero que entra en testing. (después ciclo 1, ciclo 2, sucesivamente hasta que se apruebe el producto)
- Idealmente son dos ciclos.
- **ESTRATEGIA DE PRUEBA**
    - Sin regresión: es la que más se usa (más rápida, más económica)
    - Con regresión: es mejor que la sin regresión, porque cada error que se corrige, introduce 3.

En un sw, hay requerimientos funcionales y no funcionales. Se tienen que probar los dos.

- Los funcionales son los más fáciles de testear.

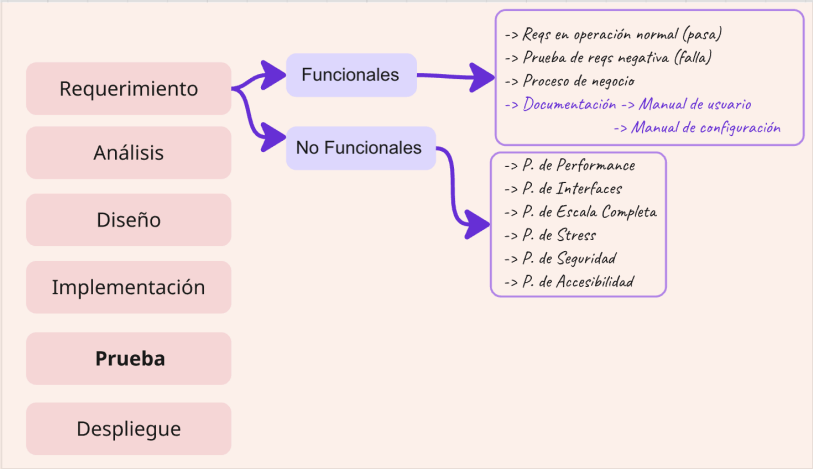
TIPOS DE PRUEBA

Describen cual es el foco de lo que querés testear con eso

Prueba de documentación: que este actualizado con la versión del sw actual

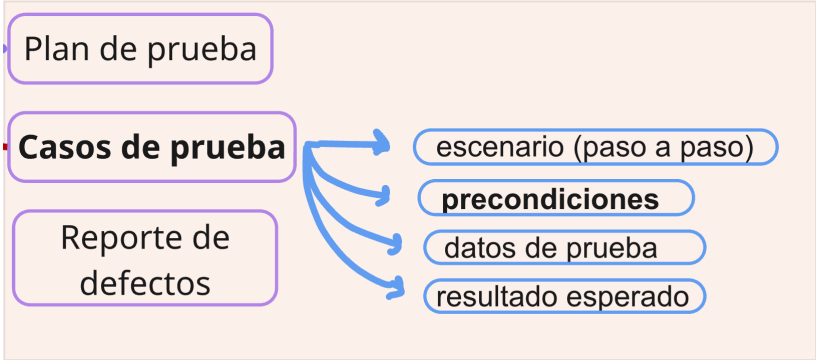
Cobertura: define del 100% de cosas que tenes que probar, cual es el porcentaje que vos si probaste

- compromiso entre lo ideal y lo posible.



- de interfeces: visual
- de escala completa: espero que pase. (evalúo el rendimiento)
- de stress: para ver en que estado queda el sistema, porque se va a caer. Veo cuanto me cuesta volver a ponerlo en línea.

CASO DE PRUEBA: es una situación descrita detallada que apunta a hacer verificación, validación de una determinada funcionalidad del sistema



necesito **precondiciones** o **condiciones de seteo** (parte de los datos) (saber que situaciones tengo que tener disponible. Datos ESPECIFICOS)

¿Cuando dejamos de probar?

- Este criterio se define en el plan de prueba
- NO cuando terminemos de encontrar errores (porque nunca terminan)
- 1. adivinación de defectos: se establece un nro donde se va a estar satisfecho
- 2. se le asigna al testing la cantidad de horas sobrantes, y lo que se encuentre en ese tiempo. es lo que se arregla (fieraso)
- 3. Se le van asignando a la nota de release, los errores que se encuentran y que no se van a librera.

Mito: “el propósito del testing es asegurar la calidad del producto” —> ES mentira!!!

“ el testing asegura que no hay más defectos” —> ES mentira !!

LA CALIDAD ES UNA DECISIÓN: HICISTE LAS COSAS BIEN O NO. NO SE AGREGA AL FINAL, en TESTING.
TESTING HACE CONTROL DE CALIDAD, NO ASEGURAMIENTO DE CALIDAD (eso es QA)