

clase grabada 4K3.

## Testing en ambientes ágiles.

- se realiza de manera continua.
- son prácticas colaborativas que se centran en incorporar calidad al producto.
- testing NO es una fase, se ~~realiza~~ realiza continuamente y NO son los únicos responsables de la calidad.

tradicional.

Pruebas de software: proceso destructivo de tratar de encontrar defectos (cuya presencia se asume) en el código.  
→ más en forma tradicional.

Testing manifiesto:

↳ en términos de valores:  
① es una actividad, NO es una fase. → actividad que ~~afecta~~ ~~a~~ ~~afecta~~ el proceso.  
se hace en conjunto con el desarrollo.

② el foco está en prevenir bugs. NO en encontrarlos.

③ "entender el testing por sobre checkear"

todo lo que se verifica, se checkeda lo tenemos que automatizar y quien hace testing debería centrarse en refinamiento de historias de usuario, pruebas de usabilidad, todo trabajo que una máquina no puede hacer.

④ enfocarse en construir software de calidad y NO ~~en~~ en "romper", encontrar defectos. ↳ esto puede producir malas relaciones desarrolladores - testers.

⑤ Todo el equipo es responsable de la calidad, NO solo el tester.

Principios de testing.

- NO es una fase, es una actividad.
- todos hacen testing.
- reducir la latencia, el tiempo, del feedback.
- mantener código limpio. Corregir rápidos errores.
- reducir la sobrecarga de documentación.
- conducido por pruebas.

Prácticas concretas:

6 pruebas de unidad e integración automatizadas.

① pruebas de regresión a nivel de sist. automatizadas.

primero se hace sobre las pruebas y se basa a estos desarrollos.

③ pruebas exploratorias

volar seg. de la función del tester.

enfoque: diseño de pruebas, aprender la funcionalidad.

todo lo hacen testing, no se hace al final, es una act. que se hace de manera continua.

④ TDD → Test Driven Development: desarrollo conducido por el testing.

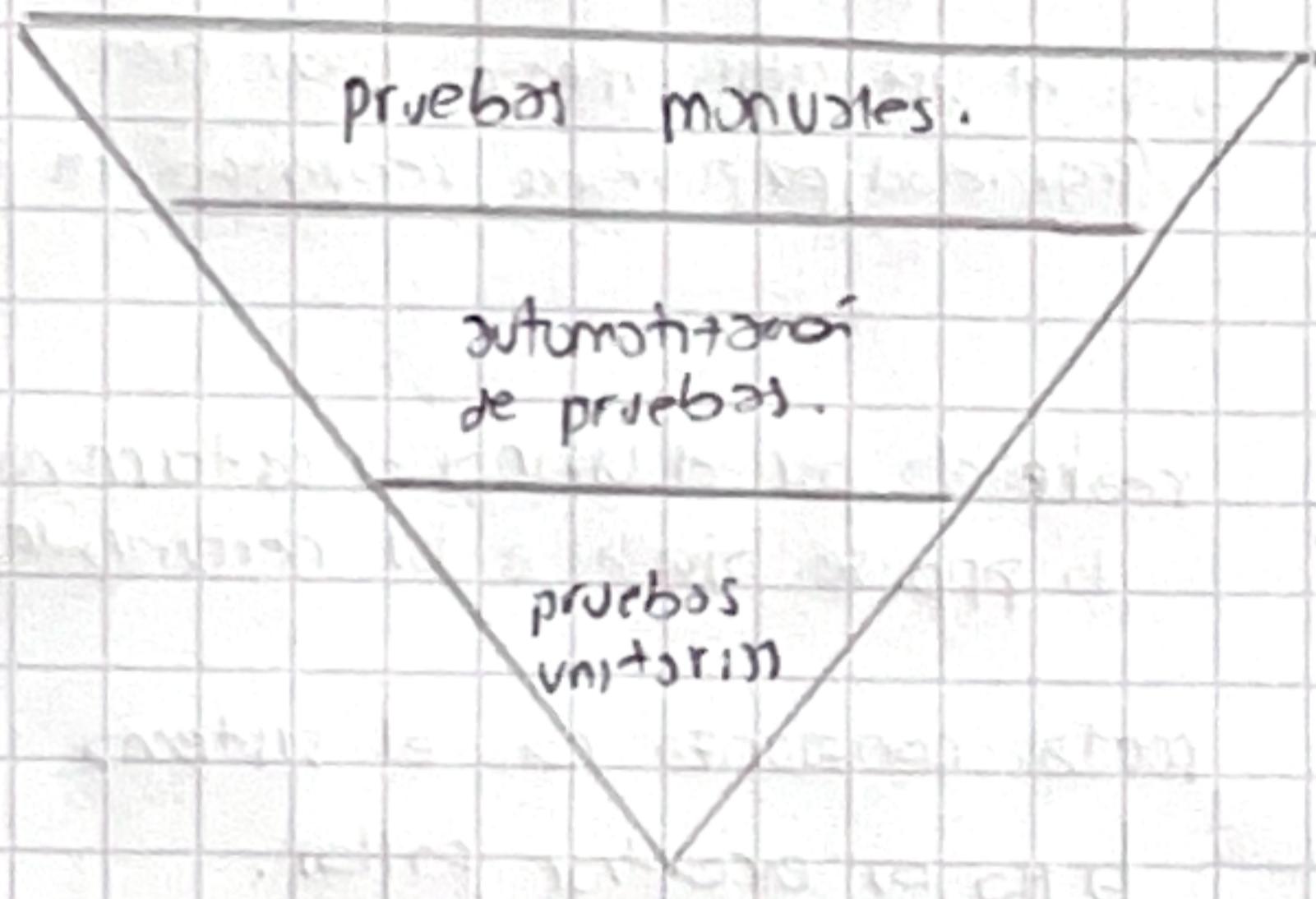
⑤ ATDD → Desarrollo conducido por pruebas de aceptación

⑥ control de versión de los pruebas con el código.

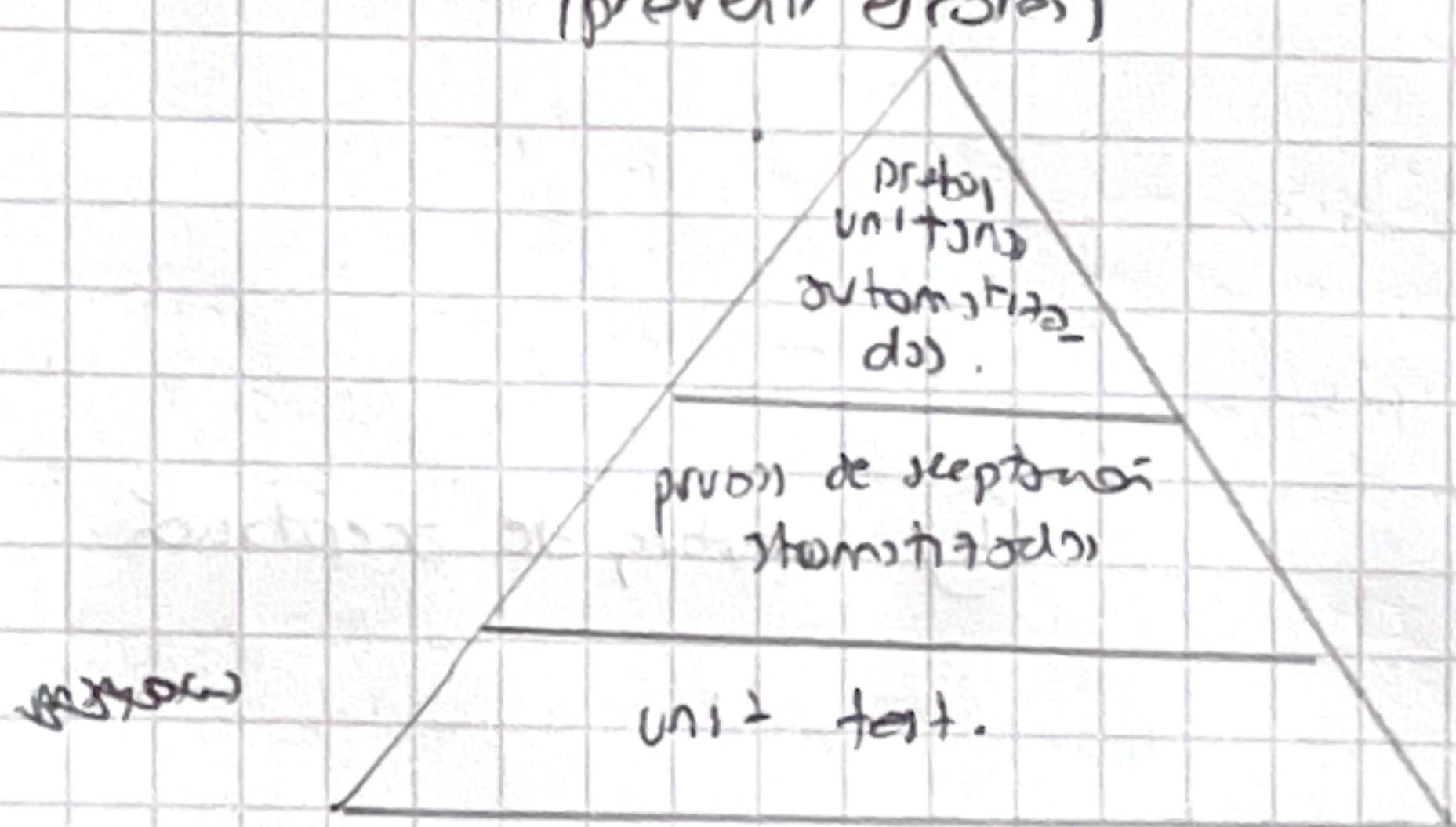
tester → ROL

Pirámide de testing:

TRADICIONAL  
(buscar bugs/detectar)



AGIL  
(prevenir errores)

Automatización en el testing

automatizar: → pruebas unitarias  
→ pruebas de integración  
→ pruebas de regresión.

Apuntes de PPT:

testing.

la calidad NO puede lograrse al final. Repetir de las tareas realizadas durante todo el ~~proyecto~~ proceso.

testing NO asegura calidad en el software ni software de calidad.

grado de severidad:

- ① importante
- ② critico
- ③ mayor
- ④ menor

⑤ contrario

prioridad

- ① urgencia
- ② alta
- ③ media
- ④ baja

niveles:

① testing unitario : x probar cada componente tras su construcción / realización.

x se prueba cada componente de manera individual e independiente.

x si se detectan errores, se reparan lo más pronto posible.

② testing de integración o de versión

x prueban que partes de un sistema que funcionan bien individualmente (probado con testing unitario) funcionan en conjunto.

x debe ser incremental → integración de arriba hacia abajo  
↳ conectar de a poco partes complejas.

↳ integración de abajo hacia arriba.

x primero probar módulos críticos.

③ testing de sistema.

x prueba cuando una app. está funcionando.  
(prueba de construcción final).

↳ si el sist. globalmente opera bien (seguridad, performance, recuperación de fail).

④ testing de aceptación

x realizado por el usuario → determina si la app. se ajusta a las necesidades.

meta: confianza con el sistema

↳ no es encontrar fallos.

Ambientes en la construcción de software:

① desarrollo

② prueba

③ pre-producción

④ producción

Caso de Prueba o set de condiciones o Variables, bajo las cuales un tester determinará si el software está funcionando correctamente o no.

+ una buena def. de casos de prueba, nos ayuda a reproducir errores).

• deben probar condiciones de prueba.

$L \rightarrow$  respuesta espontánea del sistema  
frente a un estímulo particular.

4 e) la unidad de <sup>la</sup> los actividades de la prueba.

datos de entrada y de salida: abajo a introducir.

comportamiento esperado: skills / función del Sist.  
+ esperada de award + b) requerimiento del Mimo.

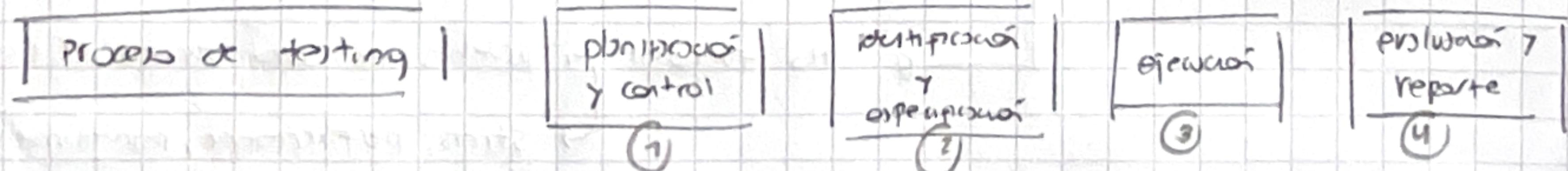
• estrategias → cosa real.

↳ coercion

ciclo de test: abarca la ejecución de los test casos de acuerdo a pruebas establecidas, aplicadas a una misma versión del sistema a probar.

reparación: al conducir un ciclo de pruebas, y reemplazar la versión del sistema sometido al mismo (cambia la versión del syst. ~ probar el nuevo), debe realizarse una revisión total de la nueva versión.

Permite detectar errores producidos por la nueva versión. Algo que ya se sabe  
puede "romperse" por algo nuevo. Ante algo nuevo, controlar todo lo anterior.



① planificar → establecer metas & testing.

parte: ① Construcción del test plan } Metros 7 Objetivo.  
estratégico.  
revisor.

⑥ Contrab { tony decide  
reviver multjobs

② disseny i pronunci de C.P.

diseño en el uso de pruebas.

NOTA Identificar datos necesarios.

### ③ desarrollo de CP.

automatizar lo necesario.

crear conjuntos de CP.

implementar y verificar el ambiente.

ejecutar CP.

revisar resultados.

comparar resultados con lo esperado.

### ④ evaluar los criterios de aceptación

reportar resultados de pruebas a stakeholders.

recibir info.

evaluar cómo resultaron las act. de testing.

#### verificación

≠

#### validación.

¿estoy construyendo el sistema correctamente?

¿estoy construyendo el sistema correcto?

porque testear?

- porque la existencia de defectos en el SFT. es inevitable. ✓
- entregar demandas del cliente ✓
- reducir riesgo ✓
- construir confianza en el producto ✓
- reducir costos por falta ✓
- verificar si el SFT. se adapta a los requerimientos y validar que las funciones se implementen bien. ✓

¿cuándo testing o supuesto?

depende de

{  
ciclos del proyecto.  
nivel de riesgo.  
criterios de aceptación

- cuando se obtiene confianza en el sistema. (funciona correctamente)
- depende del riesgo del sistema.

#### Típo de pruebas

funcional:

pruebas de función y correcta.

básica en requerimientos y en los procesos del negocio.



no funcional → pruebas de cómo funciona el sistema.

stress, performance, mantenimiento, fiabilidad, portabilidad, carga.