# Clase 05/10 Teórico

## Unidad 4: Aseguramiento de calidad de proceso y de producto

## Testing de software o Prueba de software

Lisa Crispin → Libro de Testing ágil para hacer el TIG.

https://github.com/clarabez/SoftwareTestingBooks/blob/master/Agile Testing: a practical guide for testers and agile teams (Lisa Crispin and Janet Gregory).pdf

En la clase vamos a hacer foco en el enfoque tradicional del testing.

Aseguramiento de calidad → **¿Que significa aseguramiento / garantía?**

Incorporar acciones durante el proceso de creación de software con un propósito económico. La prevención es mucho más barata que la corrección.

Conjunto de acciones para evitar los errores y evitar que esos errores se conviertan en defectos.

**Edad del defecto** → tiempo transcurre desde que cometo un error hasta que lo defecto. Mientras mayor es la edad es más costoso repararlo.

## Aseguramiento de calidad del producto

Herramienta para aseguramiento de calidad del producto → **revisiones pares / revisiones técnicas**. Alguien que está técnicamente preparado igual o mejor que yo, que no tiene una relación de jerarquía en la organización, ni es un auditor. Son la técnica por excelencia.

Las técnicas/herramientas para hacer aseguramiento de calidad:

- Inspecciones de software → revisión de pares formal
- No se como es el nombre en ingles (Recorrido) → es una revisión informal
- Auditorías → no les gusta a los enfoques ágiles

Todos los artefactos pueden ser revisados. No solamente el código.

Testing → hace control de calidad, hace validación (contra los requerimientos) y verificación (que el sistema funcione correctamente).

## Aseguramiento de calidad del proceso

**Modelos de calidad** → CMM, CMMI, ISO 9001, ISO 27000 y sus variantes (seguridad de la información)

Un modelo de calidad define un conjunto de cosas que debemos hacer, pero no definen el cómo hacerlo. Por ejemplo, te dicen que tenés que estimar pero no cómo.

Lo primero que tiene que elegir una organización es elegir un proceso que sea consistente con el modelo de calidad elegido.

Hay dos niveles:

- Se declara un proceso
- En el proyecto se controla que se respete ese proceso definido en el plan del proyecto

¿Cómo se hace? → mismas técnicas pero con otro foco.

- Auditorías (ISO) / Evaluaciones (CMMI)
- Revisiones

La calidad del producto depende de la calidad del proceso que yo uso para construirlo.

Proceso es una mesa de tres patas:

- Proceso (libro)
- Personas → capacitadas, motivadas, entrenadas y felices.
- Herramientas

# Testing de software

## ¿Qué es el testigo ágil?

- **Proceso DESTRUCTIVO de tratar de encontrar defectos (cuya presencia se asume) en el código.**

  Testing NO asegura la calidad del producto, controla la calidad (identifica defectos).

- Se debe ir con una actitud negativa para demostrar que algo es incorrecto.
- Testing exitoso → es el que encuentra defectos.
- Representa 30 a 50% del costo de un software confiable. Es la actividad más cara de la construcción de software.
- Se recomienda que lo debe realizar otra persona, excepto por el caso de las pruebas unitarias.

## Error vs Defecto

Diferencia es el momento en el que se advierte.

Error → Cuando encuentro una ~~falla~~ en la misma etapa en la que se introdujo

Defecto → Cuando la ~~falla~~ es encontrada en una etapa posterior a la que se introdujo.

**El Testing encuentra defectos**.

Los errores o defectos pueden producir o no fallas en el sistema.

## Defectos, Severidad y Prioridad

**Severidad** → cuán grave es el defecto, me ayuda a determinar el impacto del mismo.

Categorías:

1. Bloqueante → el sistema no funciona, no puedo usarlo.
2. Crítico → depende de la situación o del contexto.
3. Mayor → “”

4. Menor → “”
5. Cosmético → errores de ortografía, presentación de ciertos datos, etc.

De acuerdo a la severidad → se determinan los SLA (acuerdos de nivel de servicios)

**Prioridad** → define la urgencia del cliente para la resolución del defecto.

1. Urgencia
2. Alta
3. Media
4. Baja

No hay una vinculación uno a uno entre severidad y prioridad, depende.

# Niveles de Prueba

Existen tanto en ágil como en tradicional.

Directamente relacionado con prácticas continuas.

## Pruebas unitarias

Realizadas por el desarrollador.

TTD → construyen primero el componente probador y luego el producto.

## Pruebas de integración / Prueba de interfaces

El foco de este testing probar que varios componentes juntos funcionen entre sí.

En los libros aparece como una actividad del workflow de testing. …

## Testing de sistema o de versión

En ciclos de vida iterativos se testea un incremento. En ciclos de vida en cascada se prueba todo el producto.

## Testing de aceptación / Pruebas de aceptación de usuario

Lo realiza en el usuario.

En el PUD se hace en el workflow de despliegue.

SCRUM en la review.

# Ambientes para la construcción del software

Desarrollo →

Prueba → los desarrolladores no tienen acceso.

Pre-Producción → pruebas de aceptación de usuario. Es un ambiente costoso para la empresa. Tiene que tener los mismos recursos para simular el ambiente de producción.

Producción

# Caso de prueba

Set de condiciones o variables bajo las cuales un tester determinará si el software está funcionando correctamente o no.

Buena definición de casos de prueba nos ayuda a REPRODUCIR los defectos.

Trabajan sobre la idea de que es imposible probar todas las alternativas. De ahí surgen las técnicas que nos van a ayudar a definir casos de prueba de manera eficiente.

Los bugs se esconden en las esquinas y se congregan en los límites.

Los casos de pruebas son ejemplos específicos con datos concretos.

## Derivación de casos de prueba

De donde sacar información para crear casos de prueba:

- Desde documentos del cliente
- Desde información de relevamiento
- Desde requerimientos
- Desde especificaciones de programación
- Desde el código

## Condiciones de prueba

Definen seteo, completa un caso de prueba indicando toda la información adicional necesaria para realizar la pruebas.

Esta es la reacción esperada de un sistema frente a un estímulo particular, este estímulo está constituido por las distintas entradas.

Una condición de prueba debe ser probada por al menos un caso de prueba.

# Estrategias de prueba

## Caja blanca

- Cobertura de enunciados
- …

## Caja negra

Se definen salidas esperadas en base a determinadas entradas y se las comparan con las salidas obtenidas.

Parcial → nos dan un US y en base a eso escribimos el caso de prueba.

Métodos:

- Basados en especificaciones
  - Partición de equivalencias
  - Análisis de valores límites
  - Etc
- Basados en la experiencias
  - Adivinanza de defectos → en base a la experiencia
  - Testing exploratorio → voy a entra a probar para ver que me voy a encontrar, se hace cuando no se conoce el producto.

# Ciclo de prueba

Un ciclo de prueba es la ejecución de un conjunto de casos de prueba sobre una versión del producto.

Idealmente ejecutar 2 ciclos como mínimo.

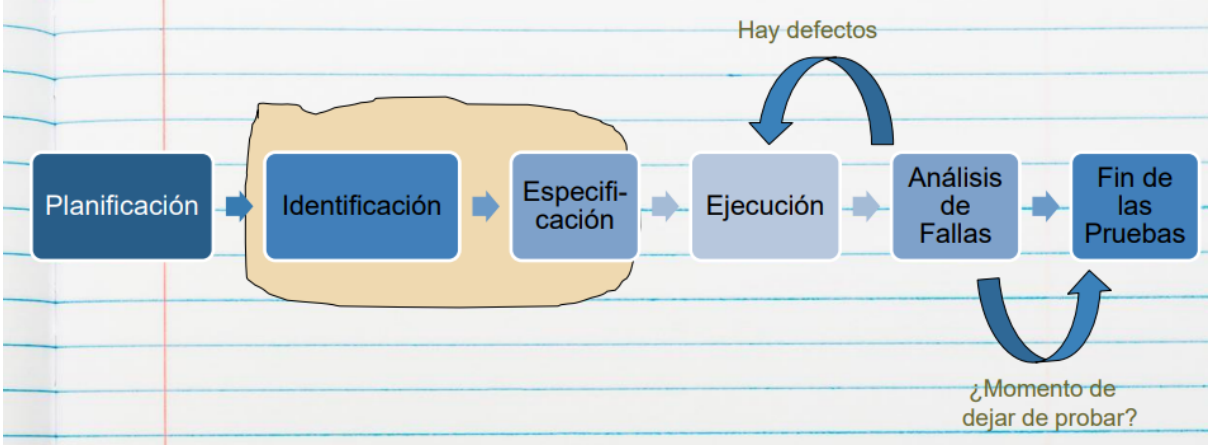
# Concepto: Regresión

- **Sin regresión** → solo probamos los defectos que se reportaron. Se pueden incorporar defectos nuevos en las funcionalidades que antes habían pasado los

casos de prueba.

- **Con regresión** → se ejecutan todos los casos de prueba de nuevo, como si fuera el ciclo 0 otra vez. Con el testing automático no hay problema en realizar testing con regresión.

## Proceso de pruebas




Planificación → que casos de prueba vamos a crear, quien es responsable, que estrategia, que metodos, de donde vamos a sacar información. A partir de que tengo los requerimientos se puede hacer.

La ejecución no se puede hacer hasta que se tiene la funcionalidad implementada.

### Entregables de testing

- plan de prueba
- casos de prueba (con los datos de prueba)
- reporte de defectos (hay que documentarlo de la mejor manera posible, capturas de pantalla y tiene asociada una severidad y prioridad
- informe final (es información para todo el equipo, de allí se obtienen todas las métricas del producto)

**¿Cuántas líneas de código debo tener para empezar a hacer testing? → 0**

# El testing en el ciclo de vida del software

## Modelo en V

Se pueden anticipar cosas/tareas de testing antes de llegar a la etapa de testing.

Se desarrolla desde lo general a lo particular y se prueba al revés.

# ¿Cuanto testing es suficiente?

El testing exhaustivo es imposible

Decidir cuánto es suficiente depende de:

- Evaluación del nivel de riesgo
- Costos asociados al proyecto

El Criterio de Aceptación es lo que comúnmente se usa para resolver el problema de determinar cuándo una determinada fase de testing ha sido completada.

Puede ser definido en términos de:

- Costos
- % de tests corridos sin fallas
- Fallas predichas aún permanecen en el software
- No hay defectos de una determinada severidad en el software

# Principios del testing

Muestra presencia de defectos

El testing exhaustivo es imposible

Testing temprano

Agrupamiento de defectos

Paradoja del pesticida → cambiar los testeadores y los casos de prueba

El testing es dependiente del contexto

Falacia a de la ausencia de errores

Un programador debería evitar probar su propio código → excepto por el testing unitario