**ASEGURAMIENTO DE LA CALIDAD**

Grado de concordancia entre los requerimientos y las expectativas que se tienen del sistema con el software logrado o resultado. Implica niveles de satisfacción del cliente, ausencia de defectos, portabilidad, etc.

Aseguramiento de la calidad

Conjunto de actividades que se llevan a cabo para evaluar y asegurar, con evidencias que el producto de software cumple con la calidad esperada. Se hace con actividades que controlan esto.



Testing entonces es la forma de verificar que efectivamente lo que se esta construyendo tiene las caracteristicas correctas.

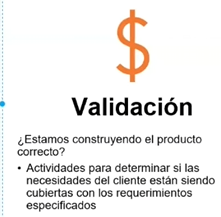
## Planificar la calidad

Ver aquellos procedimientos que voy a tener que seguir para que efectivamente se alcance la calidad que necesito tener, es decir cómo voy a controlar el software.

## Control de Calidad

es la definición y aplicación de procesos que me garantizan los procedimientos y estándares para lograr el producto que quiero.

## Existen 2 conceptos de Aseguramiento de calidad

## Tipos de actividades

### **Actividades Preventivas,**

Las **actividades** **preventivas**, evitan futuros inconvenientes por ejemplo planificando como voy a gestionar la calidad, como voy a hacer las pruebas, el hacer revisiones y auditorias

**Análisis** **causal**: aparece cuando estamos chequeando un código o una funcionalidad y nos encontramos con un problema. Ahí analizamos el por qué pasó eso y pueden haber muchas causas: gente saturada, falta de preparación, ignorancia de datos, etc.

### **Actividades Correctivas**

El testing es una actividad correctiva, hacemos pruebas de producto, verificación y validación para encontrar todos los defectos que puede haber y corregirlos.

### **Actividades Registrales**

Lo que hace es mantener la información que se fue encontrando, el registro de resultados, los informes métricas, que nos permite analizar y entender que fue lo que pasó.

TODAS LAS ACTIVIDADES TIENEN COSTOS:

* De prevención: todo los costos que conlleva prevenir antes de que pase (planificar, organizar, capacitar, testear)
* De evaluación: sirven para evaluar el nivel de calidad que tengo. Pruebas, validaciones, auditorías.
* De las fallas: implica la reparación del código.

## Pruebas

Identificar los problemas por que si no los encontramos nosotros, lo hace el cliente.

algunos conceptos es que

Un **error** es una acción humana que introduce un problema en lo que estoy generando,

el **defecto** es el desperfecto que quedo latente en el código que en algún momento cuando pase por ahi va a ocasionar una **falla** que es la manifestación física del defecto

## Objetivos de las pruebas

### **Evaluar los entregables que estamos generando**

Por ejemplo entrega de documento de requerimientos, que este bien escrito, que tenga tipo de letra correcto, bien justificado, que no tenga mal ortografía que este bien redactado.

Entregable que sea pedazo de código. Entregable que sea capacitación a usuarios finales. Verifico que el entregable tenga la condición que debe de tener.

### **Detectar Defectos**

hasta que no los fuerce a saltar con un caso de prueba no voy a ver la falla, debo detectar estos defectos

### **Cubrir todos los escenarios**

### **Reducir riesgo por software de calidad inadecuada**

### **Verificar Cumplimiento de normativas**

## Principios del testing

1. No existe ausencia de bugs

El testing demuestra que hay defectos pero no me garantiza que no hayan más, si no aparece significa que no los encontré pero no que no haya más defectos,

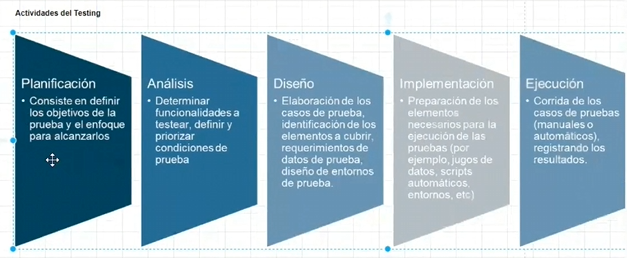
1. El testing Exhaustivo es imposible

No se puede probar todas las combinaciones posibles

1. Testing Temprano ahorra Tiempo, plata
2. Defectos se agrupan por categorías o áreas.
3. La prueba depende del Contexto. Capaz funciona en una pc, pero no en Tablet.

## Actividades del testing

Hay actividades de planificación, análisis diseño, implementación y ejecución.

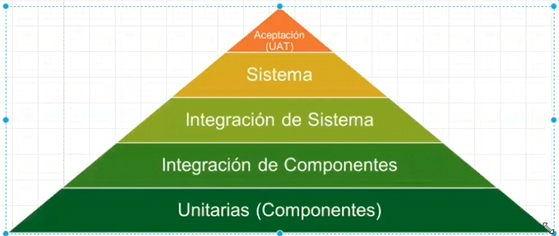


CIERRE: documentación y análisis de las actividades de prueba. Identificación de lecciones aprendidas.

## Trazabilidad

Hay que asegurarnos que todos los requerimientos estén cubiertos en algún caso de prueba, y que los resultados queden bien mapeados a que caso de prueba y que juego de dato lo ejecute. Es testing debe ser ejecutado por alguien distinto a quien construyo el caso de prueba, para ver escenarios que no pensó, o pruebe en entornos que no probó.

## Niveles de Pruebas



### **Unitarias**

Vamos a comenzar con las pruebas Unitarias, que es el testeo de cada parte por separado, por componentes. Luego debemos probar si andan cuando se fusionan.

### **Integración de Componentes**

Acá se empiezan a fusionar las cosas, por ejemplo cómo funciona el componente de alta de producto con alta de usuario, luego ver cómo funcionan ellos dos si le integro el componente de facturación.

### **Integración de Sistema**

Luego una vez que tenga todo vamos a ver cómo funciona cuando se involucran al sistema

### **Sistema**

Luego que tenemos todo integrado, vamos  a probar el sistema completo

### **UAT - User Acceptance Testing**

La última prueba, es por el usuario final.

## Tipos de prueba

Voy a tener Pruebas funcionales; enfocadas en las funciones de los requerimientos funcionales que tenemos en el sistema y Pruebas No Funcionales que testean los requerimientos no funcionales

### **Pruebas de Regresión**

Aseguran que por introducir una nueva versión no se generan impactos negativos en los que antes andaba.

### **Pruebas de Confirmación**

Son para confirmar que un defecto que corregí quedó bien corregido.

### **Pruebas estáticas**

No ejecutan el código. E**s revisar la fuente, documento, diagrama**

### **Pruebas Dinámicas**

Estamos ejecutando el código para verificar el funcionamiento, dentro de estas pruebas tenemos distintos tipos de pruebas:

#### **Prueba Caja Negra**

Son las pruebas que no me impronta como está construido el software, quiero chequear que si entro los valores tales, obtengo que los valores que recibo son lo esperado

#### **Prueba de Caja Blanca**

Se basa en conocer la estructura interna de lo que estoy testeando, se requiere conocimiento del código, entran las pruebas unitarias por ejemplo Si los if se recorren correctametne, si se ejecute correctamente el else

#### **Testing Experimental**

Se basa en la experiencia que tienen los tester al probar, dentro de testing experimental hay lo que se llama;

##### **Error Guessing**

Trato de anticipar donde van a haber errores para ir a buscarlos

##### **Testing Exploratorio**

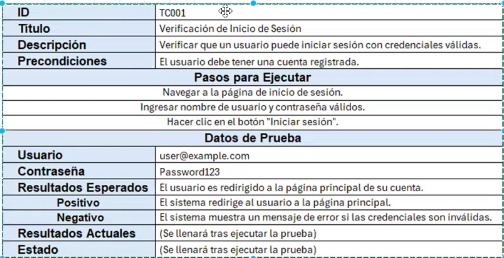
al mismo tiempo que el tester esta aprendiendo sobre el sistema sobre el objeto va a testeando sobre la marcha

### **Casos de prueba**

Son un conjunto específico de condiciones bajo las cuales se ejecutará el sistema para determinar si una de sus características funciona correctamente.

Debemos tener en cuenta:

* Descripción clara de qué se está probando
* Pasos específicos a realizar para ejecutar la prueba
* resultados esperados
* Debe ser reproducible, es decir que otra persona pueda ejecutarlo también
* En todos los casos de prueba se deben cubrir todos los requerimientos.



Resultados Positivos: Verifican que la funcionalidad se comporta como se espera

Resultados Negativos: Verifican cómo el sistema maneja condiciones incorrectas o excepcionales.

### **Plan de aseguramiento de la calidad**

dependiendo del proyecto voy a determinar cuáles van a ser aquellos objetivos, tareas que se realizarán de SQA, procedimientos específicos para cada proyecto, procesos, etc.

dentro de este plan de aseguramiento de calidad, voy a tener la:

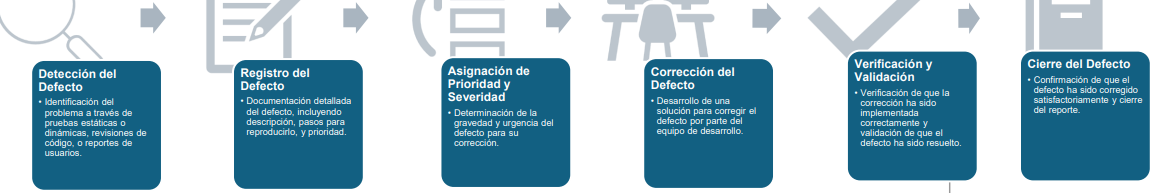
Planificación de pruebas: describe objetivos, recursos y procesos para las pruebas.

EJEMPLO de lo que debo contemplar en esta planificación de pruebas:

* **alcance** (que es lo que se cubre y lo que no se cubre de las pruebas)
* **objetivos de las pruebas**,
* **supuestos y restricciones**,
* identificar cuales son las **actividades mismas** que voy a tener que hacer (tipos de pruebas a realizar o niveles de prueba (unitarias, integración, de sistema)),
* **entorno de prueba** (entorno de desarrollo, de testing, cliente, etc),
* **secuencia de pruebas y sus niveles**,
* **criterios de inicio y fin de pruebas** (bajo qué condiciones puedo arrancar a probar, en qué entorno, con qué datos. Por ejemplo: fases previas de pruebas terminadas, juegos de datos disponibles). y bajo qué condiciones finalizo la prueba, por ejemplo: que todos los casos de prueba se hayan ejecutado.
* **criterios de suspensión de pruebas:** condiciones bajo las cuales se pueden suspender las pruebas.
* **Cronograma de pruebas:** cuales son las actividades concretas, en que fechas, quien las hace, etc
* **Plantillas**: templates, definiciones de referencia para hacer documentación, reportes.
* **conocimientos necesarios para hacer pruebas.**

**DEFECTOS**: desviaciones observadas en el comportamiento obtenido y esperado del producto. Todo aquello que no es como tenia que ser, genera un defecto.

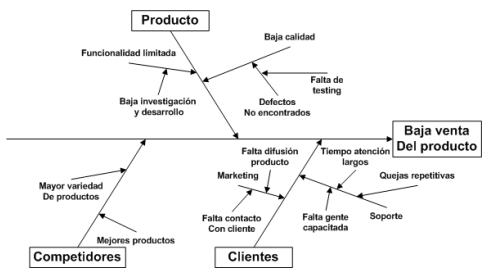
**PROCESO DE GESTIÓN DE DEFECTOS:**



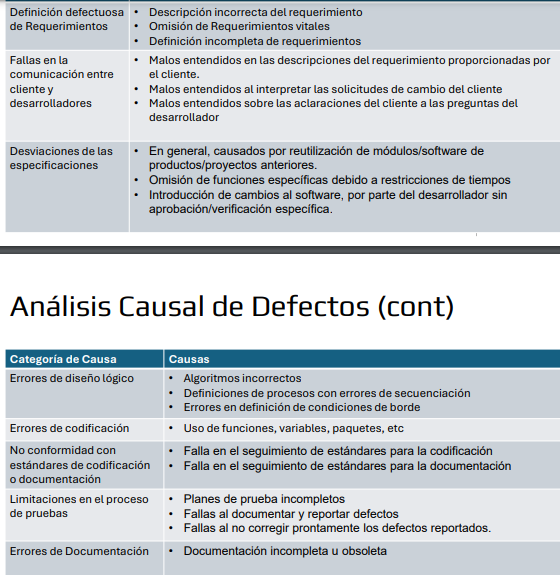
**ANÁLISIS CAUSAL:** busca identificar las causas de los problemas para erradicar los defectos que se originan en ellas, proponiendo cambios en las herramientas o procesos. Para ello se analizan los defectos que hubo. Se le pide al desarrollador que diga por qué o cómo pasó.

Se utilizan diferentes técnicas para tratar de identificar la causas:

* técnica de los 5 por qué: supone preguntar el por qué de todas los defectos u errores. es un por qué detrás de otro.
* espina de pescado: es una especia de espina de pescado donde se pone un eje. en la cabeza del pescado va el problema y a lo largo de la espina dorsal tenes áreas específicas como clientes, competidores, producto y en cada área hay espinas mas pequeñas con explicaciones del área.



CATEGORÍAS DE CAUSAS:



**Análisis de Pareto**:  Utilizado para identificar el 20% de las causas que ocasionan el 80% de los defectos. Para cada causa se cuentan cuantos defectos generó, se ordenan de mayor a menor. Luego se analizan los porcentajes de influencia en los defectos, de cada causa y se arma un plan para trabajar y mitigar eso.